FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Engenharia Electrotécnica e de Computadores

A Comparative Study between Centralization and Decentralization of Diagnostic Reasoning in EPS

Bruno Miguel Pinto Alves

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa de Lisboa para obtenção do grau de mestre em Engenharia Electrotécnica e de Computadores

Orientador: Prof. José Barata

Lisboa

2010

Aos meus pais,

à minha irmã,

à minha namorada.

AGRADECIMENTOS

Gostaria de começar por agradecer ao meu orientador, professor José Barata, por todo o apoio e disponibilidade que demonstrou ao longo da realização desta tese de mestrado. Sem a sua orientação não seria possível a concretização deste trabalho.

Um agradecimento especial ao meu colega e amigo Luís Ribeiro, por todas as sugestões dadas e ajuda facultada, na resolução de problemas que surgiram ao longo do trabalho.

Aos meus pais e irmã, pelo seu apoio incondicional demonstrado, não só durante o meu percurso académico, como também em todas as etapas da minha vida. À minha namorada, pela sua paciência, compreensão e todo o apoio que me deu.

A todos os meus colegas e amigos, Tiago, David, Filipe, Fábio, Victor, Diogo, João, Pedro, entre outros, que me possibilitaram momentos únicos e inesquecíveis durante todo este percurso académico.

RESUMO

Esta tese aborda a problemática da realização de diagnóstico em paradigmas de produção recentes, que surgem como resposta aos requisitos impostos pela evolução da sociedade. Estes sistemas são compostos por módulos inteligentes, que interagem e se coordenam entre si, de forma distribuída para a realização de uma determinada tarefa.

Estes módulos seguem o conceito de "*Plug and Play*", o que significa que devem ser capazes de se auto-organizarem, sem recorrer a qualquer tipo de reprogramação, de forma a conseguirem responder aos requisitos de produção.

Estes novos sistemas de produção distribuídos providenciam um aumento na flexibilidade e na agilidade da linha de produção, oferecendo deste modo, o suporte necessário para a produção personalizada.

Os sistemas de diagnóstico são, no contexto destes novos paradigmas, vistos como essenciais não apenas para monitorar e diagnosticar falhas, mas também, como um meio eficaz de prevenção e manutenção a curto e a longo prazo.

No entanto, devido à complexidade e à dinâmica ao nível das interacções entre os módulos, torna-se particularmente difícil implementar processos eficazes de diagnóstico, que permitam identificar e compreender as falhas entre módulos. A natureza das interacções entre os módulos vai para além da lógica de controlo, fazendo com que os sistemas de diagnóstico tradicionais, habitualmente centrados nos equipamentos e não nas relações entre estes, não se ajustem a estes novos paradigmas.

Nesta tese é feita uma comparação entre dois sistemas de diagnóstico que seguem abordagens diferentes (uma centralizada e outra distribuída). Para realizar e validar esta comparação, é definida e utilizada uma métrica de comparação que relaciona a performance dos sistemas de diagnóstico com a complexidade da rede de interacções formada pelos vários módulos.

ABSTRACT

This thesis addresses the problem of performing diagnosis in recent production paradigms that appear in response to the requirements imposed by the evolution of society. These systems are composed by intelligent modules, which interact and coordinate with each other in a distributed fashion to perform a particular task.

These modules follow the concept of "Plug and Play", which means they must be able to organize themselves, without any kind of reprogramming, so that they can meet the production requirements.

These new distributed production systems increase flexibility and agility on the production line, thus providing the necessary support for custom production.

The diagnostic systems are in the context of these new paradigms, seen as essential not only to monitor and diagnose faults, but also as an effective means of prevention and maintenance in the short and long term.

However, due to the complexity and dynamics at interactions level between modules, it becomes particularly difficult to implement effective procedures for diagnosis, to identify and understand the faults between modules. The nature of the interactions between modules goes beyond the logic control, making the traditional diagnostic systems, usually focusing on equipment rather than the relationships between them, not the suitable solutions to these new paradigms.

This thesis provides a comparison study of two diagnostic systems that follow different approaches (one centralized and other distributed). To implement and validate this comparison, is defined and used a metric that relates the performance of diagnostic systems with the complexity of the interactions network formed by several modules.

LISTA DE ACRÓNIMOS

ANN	Artificial Neural Networks
BMS	Bionic Manufacturing Systems
EAS	Evolvable Assembly Systems
EPS	Evolvable Production Systems
FDI	Fault Detection Identification
FMS	Flexible Manufacturing Systems
JADE	Java Agent DEvelopment Framework
JUNG	Java Universal Network Graph Framework
HMM	Hidden Markov Model
HMS	Holonic Manufacturing Systems
MAS	Multi-Agent Systems
QTA	Qualitative Trend Analysis
RMS	Reconfigurable Manufacturing Systems
SOA	Service Oriented Architectures

ÍNDICE DE CONTEÚDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

1.1. CONTEXTO

Os sistemas naturais muitas vezes exibem uma tendência intrínseca para se auto-organizarem em reacção a uma turbulência. Tipicamente, esse comportamento vai além dos padrões normais de acção/reacção e envolve processos internos complexos e não triviais do sistema, que podem induzir a alterações estruturais temporárias ou a longo prazo.

Uma característica fundamental destes sistemas biológicos e físicos, é o seu equilíbrio e permanente feedback com o meio ambiente, que garante o seu comportamento normal [1]. O surgimento da ordem, na natureza, é um fenómeno que é atribuído a um conjunto de combinações de interacções locais simples.

Este conceito de combinação de funcionalidades tem sido amplamente explorado em vários paradigmas de produção orientados a módulos: "*Bionic Manufacturing Systems*" (BMS) [2], "*Holonic Manufacturing Systems*" (HMS) [3][4][5], "*Reconfigurable Manufacturing Systems*" (RMS) [6][7], "*Evolvable Assembly Systems*" (EAS) [8] e "*Evolvable Production Systems*" (EPS) [9][10].

À parte das respectivas especificações, os blocos de construção destes paradigmas são apresentados como entidades independentes que fornecem uma ou mais funcionalidades. Existe um mecanismo que permite a estes indivíduos interagir, através do próprio ambiente [11], ou utilizando qualquer outra forma explícita de cooperação/colaboração. Através destas relações temporárias emergem funcionalidades, que resultam de um conjunto de interacções e de uma combinação de contribuições individuais.

Prevê-se que uma abordagem modular aos sistemas de produção, inspirada por estes mecanismos naturais e com uma fina granularidade, faça aumentar a robustez, flexibilidade, resposta e sustentabilidade dos sistemas.

No entanto, o desenvolvimento de sistemas modulares utilizando módulos distintos, que explorem estes comportamentos emergentes, apresentam desafios significativos.

Dada a natureza dinâmica e interactiva destes sistemas, bem como a frequência e a natureza da informação trocada, as falhas são susceptíveis de se propagarem, inesperadamente, através da multiplicidade de interacções desenvolvidas entre os componentes do sistema. Isto torna o processo de diagnóstico particularmente difícil de implementar.

Os métodos convencionais de diagnóstico e os instrumentos tipicamente aplicados em ambientes de produção, dificilmente, poderão ser aplicados em sistemas com uma estrutura evolutiva. Um dos principais problemas é o desenvolvimento de um modelo consistente, que idealize evolução e adaptação.

A presente tese apresenta e compara duas abordagens que exploram a dimensão da rede, no contexto de diagnóstico. Os métodos propostos diferem, principalmente, na natureza da informação (global e local) e na filosofia do diagnóstico. Em ambos os casos, a lógica de produção/controlo segue os paradigmas dos EPS.

No primeiro caso, o diagnóstico é centralizado numa entidade que usa lógica temporal para captar a natureza da propagação das falhas e, quando a falha estabiliza, tenta explicar o evento utilizando informação anteriormente aprendida, a partir do sistema, sobre a propagação de falhas.

No segundo caso, cada módulo usa um algoritmo interno para actualizar o seu estado, utilizando a sua informação sensorial e a percepção e julgamento sobre o estado dos seus vizinhos directos. O mecanismo é, necessariamente, assíncrono e todos os módulos ajustam o seu estado à medida que a falha se propaga, de uma forma totalmente descentralizada e autónoma. A implementação desta segunda abordagem não foi realizada pelo autor desta tese e como tal não será apresentada, podendo ser consultada em [89].

No contexto da dimensão da rede, é proposta e utilizada uma métrica de comparação que relaciona a performance do diagnóstico, identificação da origem da falha e a sua propagação, com um índice de complexidade inerente à rede de módulos e suas interacções.

1.2. ORGANIZAÇÃO DA TESE

Esta tese está organizada em seis capítulos: Introdução, Estado da arte, Arquitectura, Implementação, Testes e resultados e, por fim, Conclusões e Trabalho Futuro.

Este capítulo sintetiza o problema sobre investigação e apresenta o modo como está organizada a tese.

No segundo capitulo, Estado da Arte, são apresentados vários paradigmas de sistemas de produção com foco nos EPS. São também apresentadas as principais características que um sistema de diagnóstico deve apresentar, e é apresentada a métrica de complexidade a utilizar na comparação de sistemas de diagnóstico.

As arquitecturas dos dois sistemas de diagnóstico e as diferenças entre estas, são apresentadas no capítulo denominado Arquitectura, e no capítulo de Implementação são descritos os detalhes de implementação do diagnóstico que segue a abordagem centralizada.

No capítulo seguinte, Testes e resultados, é apresentada a plataforma utilizada para testar os sistemas de diagnósticos, assim como os testes realizados e os resultados obtidos.

Por fim, são apresentadas as conclusões da comparação entre os dois sistemas e as perspectivas de trabalho futuro.

2. ESTADO DA ARTE

Ao longo deste capítulo será feita uma revisão sobre a evolução da indústria de manufactura e os novos paradigmas que surgiram na sequência dessa evolução. A importância do diagnóstico, as suas características e as várias metodologias existentes na sua aplicação serão apresentadas.

2.1. SISTEMAS DE MANUFACTURA

2.1.1. EVOLUÇÃO

No decorrer dos últimos anos, o mundo tem evoluído de uma forma exponencial. Novas tecnologias surgem num piscar de olhos, impondo às indústrias de produção a necessidade de evoluírem e se adaptarem às mudanças. Esta evolução e adaptação tem de abranger, obrigatoriamente, a forma como os produtos são desenhados, produzidos, distribuídos, o ciclo de vida dos próprios produtos, assim como, a matéria-prima utilizada para o fabrico dos produtos.

Até ao inicio do século XX os produtos eram produzidos à medida e ao gosto do cliente, proporcionando um elevado nível personalização e satisfação do cliente. Em contrapartida, os produtos eram caros e apenas acessíveis a uma pequena percentagem da população.

No começo do século XX, deu-se inicio á produção em massa, que consistia na produção de grandes quantidades de produtos semelhantes da forma mais rápida e barata possível. A famosa frase de Henry Ford sobre o modelo T da Ford, "*The customer can have any color as long as it is black*" (O cliente pode ter qualquer cor desde que seja preta), é representativa da metodologia empregue na produção em massa.

No entanto, esta metodologia de produção era pouco flexível, não permitindo manobras de inovação para responder a possíveis mudanças de mercado, limitando de certo modo, as escolhas dos clientes.

A partir da década de 80 assistiu-se a um desenvolvimento socioeconómico e a um aumento do nível de vida, traduzindo-se num aumento do consumo de produtos mais personalizados, mais descartáveis e com um nível de vida mais curto. Os consumidores começaram a dar importância à variedade dos produtos existentes no mercado. A costumização do produto até ao momento irrelevante, passou a ter uma importância considerável nos produtos desejados pelos clientes.

Estes novos conceitos sobre os produtos, provocaram uma necessidade de evolução e inovação dos métodos empregues pelas indústrias, de modo a conseguirem produzir uma grande variedade de produtos a baixo preço, mas com qualidade para satisfazer as novas exigências do mercado. Unidades de produção mais pequenas e, consequentemente, mais flexíveis foram fundamentais para responder ao mercado, naquela que ficou conhecida como a produção em massa personalizada.

Actualmente, e face á grande competitividade existente no mercado, a sobrevivência de uma empresa depende directamente da capacidade de adaptação, evolução e reacção às exigências do mercado. O tempo e os custos necessários para desenvolver e implementar uma linha de produção sempre que é necessário produzir um novo produto são proibitivos. Processos de manutenção, prevenção, monitorização, diagnóstico e recuperação devem ser vistos como partes integrantes dos sistemas de controlo.

Nas tentativas iniciais de suportar a flexibilidade nos sistemas de manufactura, incluem-se os "*Flexible Manufacturing Systems*" (FMS), que pretendiam atingir elevados níveis de eficiência ao preverem situações de produção futuras aquando da sua implementação. No entanto, as soluções de controlo eram demasiados rígidas e não conseguiam lidar com situações imprevisíveis (provocadas pelos requisitos do mercado), impondo, assim, um grande esforço na programação, o que por si aumentava o tempo e o custo da instalação [12]. Deste modo, os FMS impunham um limite de flexibilidade e não garantiam uma resposta adequada aos requisitos. Tornou-se evidente a necessidade de introduzir o conceito de auto-organização nas unidades fabris, de modo a suportarem elevados níveis de reconfigurabilidade, agilidade e flexibilidade.

Em resposta às exigências anteriormente mencionadas, novos conceitos e paradigmas direccionados à manufactura surgiram ao longo dos últimos anos.

Os "*Bionicle Manufacturing Systems*" (BMS) [2] inspiram-se no funcionamento natural dos órgãos. Os órgãos são formados por células e suportam diferentes organismos através de interacções harmoniosas com outros órgãos. Esta aproximação pressupõe a ideia de um sistema hierárquico onde a informação navega de baixo para cima e vice-versa. O sistema baseia-se em auto organização, formando modelos nas várias camadas de modo a cumprir as tarefas desejadas.

Os "*Holonic Manufacturing Systems*" (HMS) [3][4][5] baseiam-se no conceito de holon, uma combinação da palavra grega "holos" (todo) e o sufixo "on" (parte). Os HMS aparecem na comunidade como uma resposta às mudanças contínuas dos requisitos e procuras das empresas. Nos HMS os "*holons*" cumprem tarefas através da coordenação, comunicação, cooperação e negociação, em que cada "*holon*" apresenta comportamentos simultâneos de um todo e de uma parte. Uma extensa revisão sobre os HMS e as suas características podem ser encontradas em [13] e [14].

A modularidade é vista pela indústria moderna como um aspecto chave para atingir flexibilidade e agilidade. A modularidade pode aumentar a evolução do sistema ao evitar que melhorias numa parte do sistema ponham em risco a evolução do sistema como um todo. De acordo com [15], a modularidade é uma simples consequência das aptidões de cada indivíduo, de tal forma que os indivíduos mais aptos introduzem um aumento na modularidade do sistema. Isto sustenta que os sistemas modernos de produção devem procurar adaptação através da modularidade, onde cada módulo individual do sistema tenha a capacidade de evoluir e, consequentemente, faça evoluir o sistema.

Os "*Reconfigurable Manufacturing Systems*" (RMS) [6][7] pretendem atingir um balanço entre a customização e a produção em massa. Segundo [16], as principais características dos RMS incluem: modularização, integração, flexibilidade, escalabilidade, diagnosticabilidade, etc. Estes princípios fazem com que as abordagens centralizadas não sejam adequadas, exigindo abordagens descentralizadas que dotem os sistemas com a capacidade de adicionar e remover dinamicamente componentes, permitindo a coordenação e adaptação com pouco ou nenhum esforço de programação [9].

Mais recentemente, novos paradigmas surgiram, "*Evolvable Assembly Systems*" (EAS) [8][17][18][19][20] e "*Evolvable Production Systems*" (EPS)[9][10][21].

2.1.2. EPS

O conceito de EPS é visto como um novo paradigma que abrange o desenvolvimento, manutenção e evolução de sistemas industriais. Semelhante aos HMS e RMS, os EPS e EAS baseiam-se na modularização e definem-se pela distribuição de módulos inteligentes nos sistemas industriais que assegurem integração e capacidade de organização (controlo, monitorização e diagnóstico) na presença de alterações na produção.

O princípio dos EAS assenta não só na capacidade de adaptação dos componentes do sistema, mas também na evolução dos componentes de modo a tornar os processos mais robustos. Os sistemas evolutivos caracterizam-se pelo controlo distribuído e por uma arquitectura inteligente e modularizada incluindo todos os métodos de suporte necessários.

O desenvolvimento dos EPS assenta nos mesmos princípios dos EAS, porém, os EPS consideram um nível de granularidade mais fino ao nível de "grippers", robots, sensores, actuadores, etc., que se recombinam para cumprirem determinadas tarefas.

Os EPS abordam as mudanças de produção inspirando-se em comportamentos biológicos como evolução e adaptação. Neste sentido e no contexto da manufactura, pode-se definir a capacidade de evolução ("*Evolvability*") como a habilidade dos sistemas complexos evoluírem com as contínuas mudanças dos requisitos, sejam pequenas mudanças na "hora" ou complexas e importantes transformações. Os EPS e os EAS procuram inspiração em outras áreas [22], como: inteligência artificial, teoria da complexidade, autonomia computacional, etc.

Segundo [23], existem dois princípios fundamentais nos EAS/EPS:

- "*The most innovative product design can only be achieved if no assembly process constraints are posed. The ensuing, fully independent, process selection procedure may then result in an optimal assembly system methodology.*"

Se a própria instalação do sistema de produção é considerada como um problema ao desenvolvimento do produto, então a perspectiva é a muito curto prazo e é inadequada. O objectivo passa por criar um sistema de produção que seja eficiente e sustentável, ao ponto de ser possível produzir qualquer produto sem ter preocupações ao nível da própria instalação.

- "*Systems under a dynamic condition need to be evolvable, i.e., they need to have an inherent capability of evolution to address the new or changing set of requirements.*"

Os sistemas evolutivos têm de ter a capacidade de se adaptarem dinamicamente a novos tipos de produtos e a cenários de produção. Em ambientes dinâmicos como os de hoje, explorar os efeitos positivos e evitar os efeitos adversos dos comportamentos que emergem do sistema é fundamental.

A introdução do conceito de auto-organização permite que um determinado sistema, durante o seu funcionamento, altere a sua configuração de uma forma autónoma sem existir qualquer tipo de controlo centralizado ou hierárquico. Os motivos da implementação de conceitos de auto-organização nos EPS são descritos em [22].

Deste modo, os EAS/EPS apresentam-se como uma solução, oferecendo uma rápida configuração da plataforma de produção, através dos conceitos referidos em [19] e enumerados de seguida:

- Módulo – Qualquer unidade que conseguir realizar uma operação e integrar uma interface específica. Os níveis de granularidade devem ser definidos (o nível menor e o maior grau).
- Granularidade – O menor nível de granularidade de um módulo, dentro de uma arquitectura de referencia, será um suporte ou uma "gripper". O maior será se uma "gripper" consegue comunicar com um robot.
- Conectividade – A habilidade de reencontrar e integrar componentes do sistema, através de uma plataforma de trabalho de uma dada arquitectura.
- Configurabilidade (Interoperabilidade) – Habilidade para encontrar novos componentes no sistema para realizar novas, mas pré-definidas, operações.

- Evolução – Sistema completamente reconfigurável que exibe comportamentos emergentes, que introduzem novos ou redefinidos níveis de funcionalidade. Sendo necessário uma arquitectura de referência rigorosamente definida para garantir a correcta aplicação das características relevantes.

Uma descrição sobre evolução dos EPS pode ser encontrada em [18] e [10].

Todos os paradigmas referidos (HMS, RMS, EAS e EPS) apresentam propriedades em comum:

- Modularidade: São considerados blocos de construção distribuídos.
- Autonomia: Cada bloco é criado independentemente dos outros e fornece determinadas funcionalidades.
- Interacção: Os módulos interagem entre si consoante determinadas regras ou acordos.
- Estrutura: Todos os blocos desempenham um determinado papel no sistema.
- Dinâmica: A maior parte dos paradigmas suporta a ideia de evolução, introduzindo alterações no sistema e capacidade de aprendizagem.
- Heterogeneidade: Os sistemas alvos destes paradigmas são composições de unidades distintas.

As diferenças entre os HMS e os EPS, e entre os RMS e os EPS são discutidas em [9] e [24] respectivamente.

Do ponto de vista tecnológico, estes novos paradigmas introduzem novos desafios e requisitos. Os "*Multi-Agent Systems*" (MAS) [25][26] e os "*Service Oriented Architectures*" (SOA) posicionam-se actualmente como as melhores ferramentas de suporte para as novas plataformas, já que suportam por omissão as características acima detalhadas e asseguram a interoperabilidade e integração necessária em ambientes heterogéneos.

O controlo industrial distribuído e descentralizado utilizando os MAS e os SOA, é actualmente uma área de interesse em desenvolvimento a nível académico e industrial [27].

Como projectos académicos pode-se destacar: ADACOR [28] – aproximação holónica para o controlo em manufactura, focada na definição de operações e em operações de manutenção, ABAS [29] – aproximação baseada em agentes que modela operações de manufactura (montagem) e por fim, COBASA [30] – sistema multi-agente para melhorar a reconfiguração e a agilidade a nível da linha de montagem.

Como projectos europeus que visam o controlo distribuído podemos destacar: EUPASS [31] – desenvolvimento de sistemas evolutivos a nível de linhas de montagem, SODA [32] – criação de um sistema orientado a serviços tendo como base a plataforma DPWS desenvolvida no projecto SIRENA [33] e SOCRADES [34] – desenvolvimento de uma arquitectura orientada a serviços (SOA) para sistemas de automação.

Um estudo sobre as características e a utilização dos MAS e SOA como tecnologias de suporte aos EPS é feito em[35]. As tecnologias (MAS e SOA) não são incompatíveis, em [36] é definida uma relação entre as duas tecnologias: O agente é a peça concreta de software ou hardware que envia e recebe mensagens, enquanto o serviço é um recurso caracterizado como uma funcionalidade que é providenciada. Deste modo, os SOA não devem excluir o uso de MAS, assim como os MAS não devem ignorar os benefícios provenientes do uso e implementação de serviços.

2.2. DIAGNÓSTICO

Com a evolução dos sistemas de controlo, em muito devido ao controlo computacional, desde cedo se percebeu que a monitorização e o diagnóstico de falhas teriam um desempenho fundamental na indústria de produção. O aumento da exigência da qualidade dos produtos, a procura da redução nos custos de produção e manutenção e o desenvolvimento de sensores e actuadores, tiveram um papel preponderante no desenvolvimento e utilização de metodologias de diagnóstico.

A detecção e o diagnóstico de falhas assumem-se, cada vez mais, como um importante campo de pesquisa na Engenharia. A detecção antecipada de falhas em áreas cruciais como aeronáutica, nuclear, naval, industrial, medicina, etc., pode fazer a

diferença evitando desastres ambientais, a perda de vidas humanas, materiais, entre outros.

Estatísticas industriais mostram que, apesar dos grandes acidentes a nível industrial serem pouco frequentes, quando estes ocorrem podem ter grande impacto financeiro, social e ambiental. É ilustrativo disso, o recente acidente numa plataforma petrolífera no Golfo do México, que para além dos prejuízos financeiros avultados, causou o maior desastre ambiental da história dos EUA. No entanto, pequenos acidentes que originam pequenas lesões e poucos danos materiais, causam prejuízos na ordem dos biliões devido à sua elevada frequência. Face a este panorama e com o objectivo de reduzir as perdas a nível da produção, sentiu-se a necessidade de definir o controlo e a gestão de eventos anormais ("*Abnormal Event Management*" – AEM) como uma das grandes prioridades [37].

Nos últimos anos, e em parte devido aos novos paradigmas de produção como os EPS, tem surgido novas metodologias de diagnóstico visando o diagnóstico em sistemas distribuídos, a fim de resolver problemas que não conseguem ser resolvidos por meio de abordagens centralizadas.

2.2.1. CARACTERÍSTICAS

Presentemente existe na literatura uma grande variedade e quantidade de técnicas e métodos de diagnóstico. De uma maneira geral, um sistema de diagnóstico deve apresentar as seguintes características [37]:

- Rápida detecção e diagnóstico – Refere-se à capacidade do sistema de diagnóstico detectar e diagnosticar rápida e correctamente as falhas. Para que tal seja possível é necessário haver um compromisso entre performance e robustez. Este compromisso tem como objectivo evitar que o sistema seja demasiado sensível a influências com uma elevada frequência (como o ruído), que pode levar, frequentemente, a falhas falsas no diagnóstico.
- Isolabilidade – O sistema deve ser capaz de distinguir diferentes falhas. A modelação do sistema e a forma como este lida com as incertezas são

fundamentais. Um algoritmo com um elevado grau de isolabilidade terá um fraco desempenho na rejeição de incertezas.

- Robustez – Diz respeito à capacidade do sistema manter reduzido o número de diagnóstico errados. O sistema deve ser robusto o suficiente de maneira a conseguir lidar com ruído e incerteza.
- Identificação de novas falhas – Para além de informar sobre o estado do processo, seja num caso de falha ou normal, um sistema de diagnóstico deve ter a capacidade de identificar se a falha detectada é ou não conhecida. O sistema deve ser projectado para que uma falha desconhecida não seja diagnosticada como uma falha conhecida.
- Adaptabilidade – As condições e o ambiente em que um processo se encontra a funcionar podem mudar devido a modificações no sistema ou a perturbações que possam surgir num dado instante. O sistema deve ser flexível o suficiente para que consiga lidar e corresponder correctamente após essas alterações.
- Capacidade de explicação – Refere-se à capacidade do sistema fornecer a informação sobre o motivo da falha, a sua origem e como foi propagada.
- Modelação e requisitos computacionais – O "peso" computacional do modelo não deve pôr em causa o desempenho do sistema de diagnóstico. O modelo do sistema deve estar em conformidade com desempenho pretendido. Em geral, para sistemas de diagnóstico em tempo real a complexidade do modelo deve ser reduzida ao mínimo.
- Identificação de múltiplas falhas – Um sistema de diagnóstico deve ter a capacidade de identificar múltiplas falhas e compreender as relações entre estas.

2.2.2. MÉTODOS DE DIAGNÓSTICO

A comunidade científica, responsável por investigar e desenvolver modelos de diagnóstico, não é clara relativamente á classificação das metodologias de diagnóstico. Sabendo que existem outros autores com diferentes tipos de classificação, vai-se ter como referência a classificação efectuada em [37],[38] e [39].

As duas principais características para classificar um diagnóstico são:

- Tipo de conhecimento (informação)
- Tipo de estratégia utilizada para gerar o diagnóstico

Tipicamente, a estratégia utilizada para gerar o diagnóstico depende directamente do esquema de representação do conhecimento, que por sua vez depende do tipo de informação presente no sistema. Por esta razão o tipo de conhecimento usado é a mais importante característica para distinguir os métodos de diagnóstico.

Em[37], os métodos de diagnóstico são divididos em métodos quantitativos, métodos qualitativos e métodos baseados no histórico, (Figura 1).

Os modelos baseados em métodos quantitativos baseiam-se na avaliação de inconsistências (resíduos), gerados através de relações matemáticas entre as entradas e as saídas do sistema. Ao contrário dos modelos quantitativos, nos modelos baseados em métodos qualitativos, as relações entre as entradas e as saídas do sistema, são expressas através de funções qualitativas. Em contraste com as abordagens baseadas em modelos, onde é necessário, a priori, o conhecimento sobre o sistema, nas abordagens baseadas em informação histórica é apenas necessário o acesso a uma grande quantidade de dados históricos. Existem diferentes maneiras de transformar e representar esta informação em conhecimento, a priori, para o sistema de diagnóstico. Este processo é conhecido como a extracção de características e pode ser feita de um modo qualitativo ou quantitativo [37].

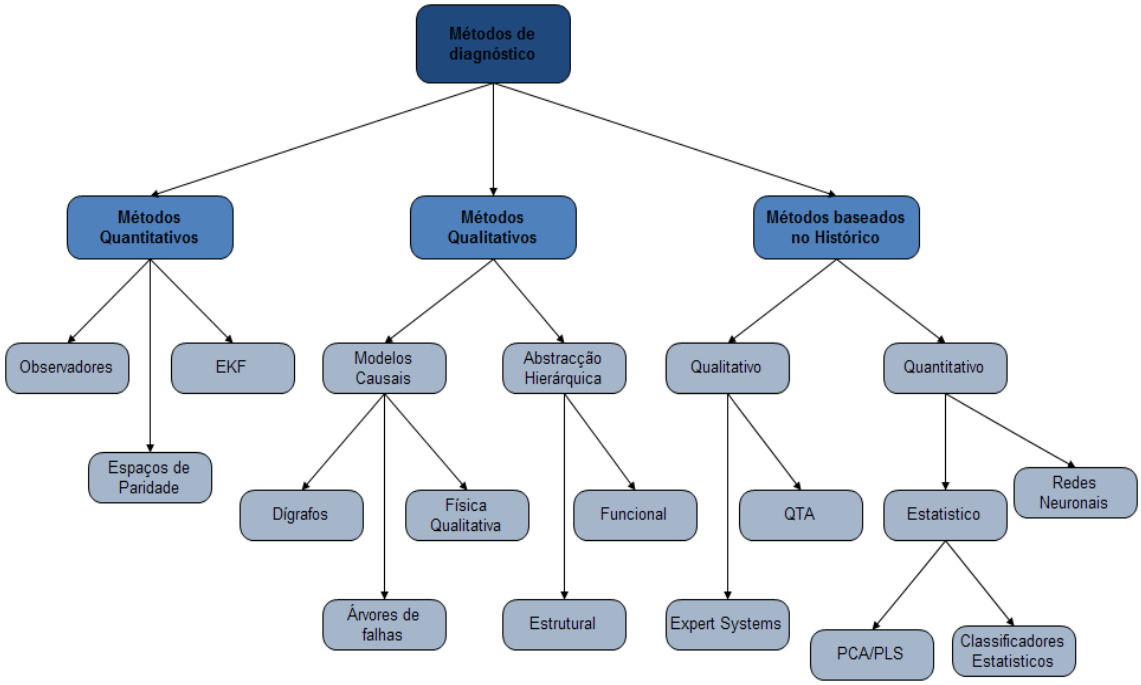

Figura 1 – Classificação dos algoritmos de diagnóstico
(Imagem adaptada de [37])

2.2.2.1. MÉTODOS QUANTITATIVOS

Na área do controlo automático, a detecção de falhas é feita através de modelos baseados em detecção e isolamento de falhas ("*Fault Detection Identification*" – FDI). Todos os modelos FDI requerem dois passos, a geração de resíduos e a escolha de regras de decisão utilizadas no diagnóstico.

Para gerar os resíduos é necessário o uso de alguma forma de redundância. Existem dois tipos de redundância, a redundância de hardware que recorre ao uso de sensores redundantes para detectar inconsistências e a redundância analítica que através de modelos matemáticos, compara o comportamento actual do sistema com o modelo previamente construído. A aplicação da redundância analítica é limitada devido ao custo extra dos sensores e ao espaço adicional necessário. Algumas das técnicas que utilizam a redundância analítica são os espaços de paridade e os observadores.

ESPAÇOS DE PARIDADE

As relações de paridade são representadas por equações obtidas através de transformações dos sinais de entrada e saída ou do estado de espaços do modelo do sistema [40]. O objectivo é verificar a paridade (consistências) do modelo através dos sensores de saída (medidas) e dos processos de entrada conhecidos. Em circunstâncias ideais, os valores dos resíduos obtidos seriam zero. Contudo, na realidade os valores resíduos nunca são zero devido ao ruído, aos erros de sensores e actuadores e às falhas do sistema.

OBSERVADORES

Os métodos de FDI baseados em observadores têm como objectivo gerar resíduos que permitam detectar e identificar diferentes tipos de falhas. Cada observador deve ser sensível a uma determinada falha e insensível às restantes. Durante o funcionamento normal, cada observador deve apenas gerar resíduos muito pequenos. Quando uma falha ocorre, todos os observadores continuam a gerar resíduos insignificantes excepto o observador sensível à falha. Cada falha deve apresentar um diferente padrão de resíduos que permita o isolamento de falhas.

FILTROS DE KALMAN

A natureza dos sistemas dificulta, frequentemente, a aplicação de modelos matemáticos. Nestas circunstâncias, os estados podem ser estimados através do uso de filtros de Kalman. Um filtro de Kalman é formado por um conjunto de equações matemáticas e proporciona um meio computacionalmente eficiente para estimar o estado de um processo linear, minimizando a média do erro quadrático. Este tipo de filtro permite estimar estados passados, presentes e até mesmo futuros, mesmo não conhecendo a natureza precisa do sistema modelado [41].

2.2.2.2. MÉTODOS QUALITATIVOS

As técnicas baseadas em modelos qualitativos utilizam, tipicamente, lógica simbólica (raciocínio simbólico), já que não existe nenhuma informação ou senso comum heurístico sobre falhas. A única informação disponível é a descrição do sistema e as observações do modelo do diagnóstico. Existem três tipos de lógica simbólica:

- Abdutiva – Gera hipotéticas explicações através das observações. Ao contrário da lógica simples, a lógica abdutiva fornece várias explicações para as mesmas observações e serve de suporte à tomada de decisões na presença de incertezas com vários pesos.
- Indutiva – Através de um conjunto de informação representativa o sistema estabelece regras para classificar (em categorias ou conceitos) essa informação.
- Default – Baseia-se em pressupostos sobre a informação que está a ser manipulada, com o objectivo de determinar nova informação que sobreponha a existente ou que rejeite a hipótese por defeito levando a uma inconsistência.

Uma informação mais detalhada sobre os tipos de lógica e suas vantagens e desvantagens pode ser encontrada em [38].

MODELOS CAUSAIS

Os modelos causais tentam captar as relações causa/efeito e vice-versa, aplicando os tipos de lógica previamente enumerados.

DÍGRAFOS

O diagnóstico tenta deduzir a estrutura do sistema a partir dos seus comportamentos, este tipo de dedução precisa de interpretar os dados recolhidos de forma a relacionar as causas e os efeitos. Estas relações podem ser representadas através de um dígrafo ("*Signed Diagraph*" – SDG) em que os arcos têm um peso positivo ou

negativo associado representando o efeito que um nó tem sobre outro As ligações representadas pelos arcos estabelecem as ligações entre as causas e os efeitos.

ÁRVORES DE FALHAS

As árvores de falhas são métodos para analisar a segurança e confiança no sistema. Geralmente são constituídas por camadas de nós e a cada nó é associada uma diferente lógica de propagação, como "*AND*" ou "*OR*". As árvores de falhas são utilizadas em várias avaliações de risco e análise de confiança, podendo também ser utilizadas para prevenir ou identificar falhas antes do seu acontecimento.

A análise de uma árvore de falhas é dividida em quatro fases distintas; definição do sistema; construção da árvore de falhas; avaliação qualitativa e avaliação quantitativa. A construção de uma árvore de falhas implica, à partida, um conhecimento aprofundado do sistema. Essa mesma construção é realizada respondendo à pergunta: "O que pode causar um evento de topo?" e através da resposta são gerados eventos ligados por nós lógicos. Uma vez que usam lógica combinatória, as árvores de falhas são um método que permite a análise das falhas.

As árvores de falhas são um método que provém dos dígrafos [42], e uma vez que não servem para detecção de falhas, também não são adequadas para sistemas de diagnóstico.

FÍSICA QUALITATIVA

A física qualitativa modela, qualitativamente, as variáveis numéricas e usa álgebra específica, com operações bem definidas, para detectar alterações na análise de processos dos sinais das derivadas. Como métodos de diagnóstico que recorrem a física quantitativa pode-se referir a simulação qualitativa ("*Qualitative Simulation*" - QSIM) e a teoria qualitativa de processos ("*Qualitative Process Theory*" – QPT).

ABSTRACÇÃO HIERÁRQUICA

A abstracção hierárquica baseia-se numa decomposição hierárquica e tem como objectivo viabilizar a representação do comportamento de todo o sistema, apenas a partir das leis que regem o comportamento dos subsistemas. A decomposição de um sistema pode ser divida em dois tipos:

- Estrutural – representa a informação de conectividade do sistema e dos seus subsistemas.
- Funcional – representa as relações meios-fim ("*means-end*") entre o sistema e os subsistemas.

A decomposição estrutural é um tipo de decomposição eficiente em sistemas onde existe uma equivalência geral entre estrutura e função. O diagnóstico pode ser considerado como uma procura *"top-down"* desde o nível mais alto de abstracção onde são considerados os sistemas funcionais, até um nível mais baixo onde são consideradas apenas unidades individuais e onde funções unitárias são analisadas.

2.2.2.3. MÉTODOS BASEADOS NO HISTÓRICO

Os métodos baseados no histórico podem-se dividir em qualitativos e quantitativos. Os métodos qualitativos incluem os "*Expert Systems*" e a análise de tendência qualitativa ("*Qualitative Trend Analysis*" – QTA).

Os métodos quantitativos baseiam-se na utilização de redes neuronais ("*Artificial Neural Networks*" – ANN) e extracções estatísticas, como a análise de componentes principais, mínimo dos quadrados parciais e classificadores estatísticos [39].

EXPERT SYSTEMS

Os "*Expert Systems*" são, em geral, sistemas muito eficientes e especializados que lidam com problemas num domínio restrito. A informação é armazenada na forma de regras do tipo "*if-then-else*" que, posteriormente, são interpretadas por um motor de

inferência produzindo determinadas conclusões. As principais vantagens do uso de "*expert systems*" são o seu fácil desenvolvimento, raciocínio transparente, a habilidade de raciocínio sobre incertezas e, ainda, a capacidade de providenciar explicações para as soluções fornecidas.

ANÁLISE DE TENDÊNCIA QUALITATIVA (QTA)

A análise de tendência qualitativa pode ser utilizada para explicar os eventos de maior importância que estão a decorrer no processo, para diagnosticar falhas e ainda para prever futuros estados. A abstracção qualitativa permite uma representação compacta da tendência ao representar apenas os eventos significantes.

Para efeitos de diagnóstico, a representação da tendência qualitativa fornece informação valiosa que facilita o raciocínio sobre o comportamento do processo. Por norma, uma falha provoca uma tendência distinta através dos sensores de monitorização que pode ser utilizada para identificar a anomalia do processo.

REDES NEURONAIS

As redes neuronais tentam simular o funcionamento das células do cérebro humano (neurónios) e são um recurso computacional muito utilizado no reconhecimento de padrões. São treinadas através de um conjunto de dados que contêm toda a dinâmica do sistema que pretendemos simular. Através deste treino os pesos de entrada para cada neurónio da rede vão sendo ajustados, com o intuito de simularem correctamente o sistema.

Cada rede neuronal pode ser constituída por um número infinito de neurónios, sendo que este número resulta de um compromisso entre a correcta simulação do sistema e a especificidade da rede em relação aos dados de treino. Este compromisso é necessário uma vez que, caso a rede tenha neurónios em excesso, fica muito específica para os dados de treino perdendo capacidade de generalização. Em geral, as redes neuronais utilizadas no diagnóstico de falhas são classificadas em duas dimensões, arquitectura da rede e estratégia de aprendizagem [39].

EXTRACÇÕES ESTATÍSTICAS

Os estados futuros de sistemas não são completamente determinados através dos estados passados, presentes ou acções de controlo futuras. Esta limitação cria um problema em lidar com sistemas que apresentem distúrbios aleatórios.

A configuração do sistema, de forma probabilística, apresenta-se como uma forma de lidar com este problema. Quando o processo funciona de forma normal as observações apresentam uma probabilidade distribuída correspondente ao modo de operação normal. No caso de haver algum evento que retire o processo do seu funcionamento normal, a probabilidade distribuída assume então outros valores.

A extracção estatística a partir da informação evoluiu desde métodos de análise gráfica para sistemas com variáveis independentes, até métodos de projecção estatística de sistemas com variáveis dependentes.

A análise de componentes principais ("*Principal Component Analysis*" – PCA) e o mínimo dos quadrados parciais ("*Partial Least Squares*" – PLS), revelam-se métodos de extracção estatística de sucesso na aplicação em processos de análise e controlo e diagnóstico [39].

O diagnóstico de falha acaba por tornar-se problema de classificação e, como tal, um sistema de diagnóstico pode ser visto como um classificador estatístico de padrões. Neste sentido, o classificador de Bayes apresenta-se como um óptimo classificador, caso as classes sigam uma distribuição Gaussiana e a informação acerca das distribuições esteja disponível.

2.2.2.4. MÉTODOS HÍBRIDOS

Por vezes um único método de diagnóstico não consegue apresentar todas as características desejadas num sistema de diagnóstico. Nesses casos, utiliza-se uma combinação de métodos de diagnóstico diferentes denominada de sistema híbrido, com a finalidade de fazer face às limitações individuais de cada método.

2.3. DIAGNÓSTICO NA MANUFACTURA

Como constatado anteriormente, existem diversas técnicas de diferentes naturezas que podem ser aplicadas em sistemas de diagnóstico. As funções e as vantagens que um sistema de diagnóstico pode acrescentar na área da manutenção foram também mencionadas.

Neste sentido, um esforço significativo na pesquisa e desenvolvimento é direccionado a resolver problemas de diagnóstico específicos, provocados pelas instalações existentes mas, também, pelas novas arquitecturas e metodologias. Estas abordagens diferem, não só, pelos mecanismos de raciocínio utilizados, como também na quantidade e variedade de sensores utilizados.

Uma das áreas de desenvolvimento de diagnóstico específico em manufactura, centra-se em manipuladores robóticos industriais, como exemplos: um sistema de supervisão para monitorizar e recuperar de falhas, que usa um motor de inferência para diagnosticar e aprender falhas por indução é apresentado em [46]; uma proposta de um sistema de diagnóstico que baseia-se na análise de sons é feita em [47]; em [48] é desenvolvida uma abordagem que utiliza um observador e um modelo matemático dinâmico para manipuladores com um determinado grau de liberdade; um método para detectar falhas isoladas em manípulos cooperativos é desenvolvido em [49].

Outra das áreas que tem sido alvo de um diagnóstico específico é os motores eléctricos [50] e as máquinas rotativas [51].

Com o aumento da complexidade das instalações, os diagnósticos tiveram de ser "alargados" para conseguir abranger mais do que um dispositivo. Encontra-se na literatura várias abordagens de diagnóstico que consideram o sistema industrial completo.

Em [52], os componentes do sistema e os testes de monitorização são modelados como maquinas de estado ("*State Machine*" – SM), permitindo a criação de bibliotecas de SM, que são, rapidamente, combinadas e aplicadas em várias situações. Uma proposta para a aplicação do modelo escondido de Markov ("*Hidden Markov Model*" – HMM) e da análise dos componentes principais para diagnosticar processos químicos é feita em [53].

O uso de redes Bayesianas para diagnosticar e estudar processos em linhas de montagem é feito em [54]. Um sistema de controlo hierárquico e uma arquitectura de diagnóstico baseada em redes de Petri para optimizar a produção e garantir a recuperação de falhas é apresentado em [55]. Um supervisor de controlo e monitorização para células de manufactura é proposto em [56]. O desenvolvimento de uma abordagem híbrida que integra teoria dos grafos, "*fuzzy sets*" e algoritmos genéticos para o diagnóstico em sistemas de manufactura é descrita em [57]. Um modelo de diagnóstico hierárquico baseado na análise de árvores de falhas ("*Fault Tree Analysis*" – FTA) e dois outros modelos de diagnóstico baseados em lógica e em sequência de controlo de falhas em sistemas de manufactura são apresentados em [58].

Com os progressos obtidos na pesquisa em sistemas de controlo distribuído, o diagnóstico é visto como uma parte integrante da arquitectura. Neste sentido, as plataformas multi-agentes (MAS) e as arquitecturas orientadas a serviços (SOA) são, maioritariamente, utilizadas.

Uma solução baseada em MAS que requer a modelação estrutural do sistema e que usa vários agentes para detectar falhas é apresentado em [59]. Em [60], um MAS é aplicado em diagnóstico de veículos para resolver os elevados requisitos de uma abordagem centralizada e em [61] os agentes formam a base de um sistema tolerante a falhas que assenta em auto-organização. Uma abordagem que utiliza MAS para identificar e diagnosticar falhas num conjunto distribuído de robots heterogéneos, e fornece mecanismos de recuperação é apresentada em [62]. A análise e proposta a um modelo de diagnóstico que utiliza MAS, e em que partes incompletas do modelo do sistema são utilizadas para detectar falhas, é feita em [63] e [64].

Através de todas as soluções anteriormente mencionadas, fica claro que realizar o diagnóstico num sistema industrial é uma tarefa complexa e não existe nenhuma ferramenta única para tal. Ao diagnosticar um sistema, aparecem problemas críticos como a propagação de falhas, localização de falhas, problemas de interoperabilidade, etc. Para além disso, a maioria do equipamento industrial apresenta limitações de processamento. Existem poucas e, por vezes, nenhumas interfaces de comunicação para sistemas computacionais, o que restringe a aplicação de sistemas inovadores de controlo e diagnóstico.

2.4. DIAGNÓSTICO EM EPS

Os EPS são sistemas complexos que providenciam novos desafios e oportunidades no campo do diagnóstico. A interacção e propagação de falhas são mais difíceis de detectar e torna-se, também, mais difícil criar um modelo de diagnóstico para um sistema, que no limite, pode estar em mudanças constantes e pode ser heterogéneo.

A maior parte dos sistemas de diagnóstico são desenvolvidos tendo em conta uma determinada linha de montagem, ou os seus subsistemas, e mesmo plantas inteiras do sistema de manufactura em vez dos seus componentes individuais. Assim, os métodos de diagnóstico tradicionais requerem, na maior parte dos casos, o modelo completo do sistema e não são directamente aplicáveis aos paradigmas de sistemas de produção modernos como os EPS.

Os sistemas clássicos de diagnóstico baseiam-se num cérebro central com uma elevada capacidade de processamento que recebe toda a informação do sistema. Num ambiente de rede, este tipo de sistema pode ser afectado devido à performance da rede, o que pode provocar atrasos na recepção da informação e, consequente, perda de reactividade às falhas. No entanto, a inteligência com base em informação local pode ser explorada para contornar este problema.

À medida que os novos sistemas de controlo são desenvolvidos, torna-se evidente a necessidade de incluir capacidades de diagnóstico nos sistemas de produção. Paradigmas como RMS [16] e EPS [21] prevêem as acções de diagnóstico como uma parte essencial e crucial nos futuros sistemas de produção.

Um sistema de diagnóstico para os EPS deve apresentar características como: detecção e diagnóstico antecipado de falhas, robustez, identificação múltipla de falhas, descriminação das falhas, etc.

O tipo de natureza dos EPS tem um impacto directo no diagnóstico em EPS. As entidades têm de ter a capacidade individual de monitorizar, diagnosticar e recuperar de falhas, ao mesmo tempo que estas, resultantes das interacções, são detectadas. Para detectar falhas resultantes de interacções, pode ser utilizada uma camada superior com a capacidade de monitorizar e validar as acções de conjuntos de entidades. A diferença entre as duas camadas, é que a de baixo nível utiliza modelos sensoriais e a de mais alto

nível utiliza modelos de interacções. Através deste segundo nível de diagnóstico, torna-se possível detectar e explicar múltiplas falhas.

Uma análise a curto prazo, serve para detectar e explicar falhas catastróficas, como falhas mecânicas, colisões, etc. A médio e longo prazo é possível, através das capacidades dos EPS, prever futuras falhas resultantes do desgaste do material, poeira, etc. Este tipo de análise garante robustez ao sistema e torna possível a detecção antecipada de falhas.

Um sistema de diagnóstico direccionado aos EPS, tem também de ter a capacidade de evoluir e adaptar-se às mudanças, caso contrário o conceito de EPS não é aplicável.

Resumindo, para preservar e manter os benefícios evolutivos, inteligentes, distribuídos, robustos e flexíveis de um sistema EPS, a camada de diagnóstico deve ser evolutiva, proactiva e reactiva a falhas Deve ter a capacidade de fazer análise preventiva, identificar falhas resultantes de interacções e deve acima de tudo preservar todas as suas capacidades de diagnóstico mesmo quando uma parte da infra-estrutura de diagnóstico não está a trabalhar.

2.5. COMPARAÇÃO DE SISTEMAS DE DIAGNÓSTICO EM EPS

Os modelos de diagnóstico, que utilizam métodos tradicionais, são normalmente aplicados em sistemas onde se sabe previamente qual o seu comportamento. Nestas circunstâncias, torna-se relativamente fácil analisar e comparar a performance de vários modelos de diagnóstico, visto não haver alterações significativas no comportamento do sistema.

Nos novos paradigmas de produção modernos, como os EPS, são definidas interacções dinâmicas entre os vários componentes do sistema. A natureza e a quantidade destas interacções podem, na maior parte dos casos, introduzir no sistema uma elevada complexidade [71]. Como os EPS contemplam evolução e adaptação, a complexidade de um sistema não é constante, ao contrário dos sistemas de produção tradicionais.

Face a isto, a modelação de um sistema numa perspectiva de rede, para compreender as interacções e consequentemente diagnosticar as falhas entre componentes, torna-se deveras interessante e promissor.

Observando o sistema do ponto de vista de rede, formada pelas interacções entre os componentes, a complexidade desta assume um papel determinante na possível propagação de falhas entre os componentes. Uma elevada complexidade pode, no pior dos casos, originar uma explosão catastrófica na propagação das falhas.

Neste sentido, a complexidade de um sistema pode ter um impacto directo nos resultados obtidos pelos modelos de diagnóstico, e como tal deve ser tida em conta na análise ao desempenho dos sistemas de diagnóstico e também na comparação entre modelos de diagnóstico.

A complexidade de uma rede pode ser obtida através da matriz de adjacência e da matriz de distâncias da rede do sistema (formada pelos seus componentes e interacções) [75], e tem a seguinte expressão:

$$C_{Sistema} = \frac{< a_i >}{< d_i >} = \frac{\frac{A}{V}}{\frac{D}{V}} = \frac{A}{D} \quad (1)$$

Em que, $< a_i >$ é a conectividade média da rede, calculada através da soma dos elementos da matriz de adjacência, e $< d_i >$ é a distância média entre os vértices da rede, calculada pela soma dos elementos da matriz de distância da rede.

No entanto, esta métrica de complexidade pode, nas redes em que as interacções têm sentidos, provocar erros no cálculo da complexidade.

Se um componente não tiver interacções com o sentido de entrada, o equivalente a um componente não poder ser afectado por uma falha com origem em outros, a distância a esse componente, por outro qualquer, é infinita o que pode levar a uma métrica de complexidade desajustada.

Face ao problema anteriormente mencionado, e visto que podem haver componentes sem nenhuma interacção de entrada num sistema EPS, a equação indicada em (1) para definir a complexidade de uma rede não pode ser utilizada.

Como índice de complexidade da rede, pode ser utilizada a conectividade média da rede, que representa o número médio de vizinhos de cada componente da rede. Assim, a complexidade da rede pode ser obtida através da matriz de adjacência da respectiva rede, e é dada pela seguinte equação:

$$C_M = \frac{\sum_1^n \sum_1^p Adj_{n,p}}{n} \quad (2)$$

Em que, n e p têm o numero de componentes da rede e $\sum_1^n \sum_1^p Adj_{n,p}$ indica o valor, 0 ou 1, da matriz de adjacência na posição (n, p).

Deste modo, através de C_M, torna-se possível analisar a performance de um sistema de diagnóstico perante a complexidade da rede de componentes à qual se está a aplicar o diagnóstico.

3. ARQUITECTURA

Neste capítulo serão apresentadas as arquitecturas dos dois sistemas de diagnóstico propostos, mais concretamente a abordagem centralizada e a abordagem distribuída. É ainda elaborada uma tabela contendo as principais diferenças entre as duas arquitecturas e em que medida é que são comparáveis.

3.1. DIAGNÓSTICO CENTRALIZADO

O sistema de diagnóstico centralizado tem como objectivo identificar a origem de uma falha e as suas propagações, através da análise de informação proveniente da área circundante onde foi registada a falha. Como tal, o sistema de diagnóstico deve possuir a informação de todos os dispositivos do sistema a que se pretende aplicar o diagnóstico, de forma a conseguir reunir a informação necessária.

O mecanismo de diagnóstico utiliza o conhecimento extraído de falhas passadas para diagnosticar as novas. Este conhecimento representa a informação histórica sobre a propagação de falhas entre os vários componentes do sistema.

Na presença de uma falha, o sistema de diagnóstico tenta identificar padrões na propagação de falhas, ao relacionar a informação extraída das falhas passadas com o estado actual do sistema.

O processo de diagnóstico deve assimilar possíveis alterações na infra-estrutura onde está aplicado o sistema de diagnóstico, ou seja, a introdução de novos elementos do sistema não requer nenhuma configuração prévia do sistema em termos de diagnóstico. Esta propriedade do sistema de diagnóstico é crucial para poder ser aplicado a EPS, o diagnóstico torna-se assim dinâmico e evolutivo, apresentado a capacidade de acompanhar a evolução do sistema a diagnosticar.

Para tal, é fundamental que o sistema possua a capacidade de avaliar e actualizar, se necessário, a informação extraída das falhas. Este processo de avaliação, é feito através da comparação da informação passada, com a informação que o sistema obtém

sempre que realiza um diagnóstico. Deste modo, a informação guardada sobre a propagação de falhas, é constantemente testada assegurando-se que esta continua representativa das falhas provenientes do sistema a diagnosticar.

Na Figura 2 encontra-se a representação da arquitectura proposta para o sistema de diagnóstico centralizado, contemplando as propriedades dinâmicas do tipo de sistema alvo a diagnosticar (EPS), o método de diagnóstico apresentado e a capacidade de aprendizagem referida anteriormente.

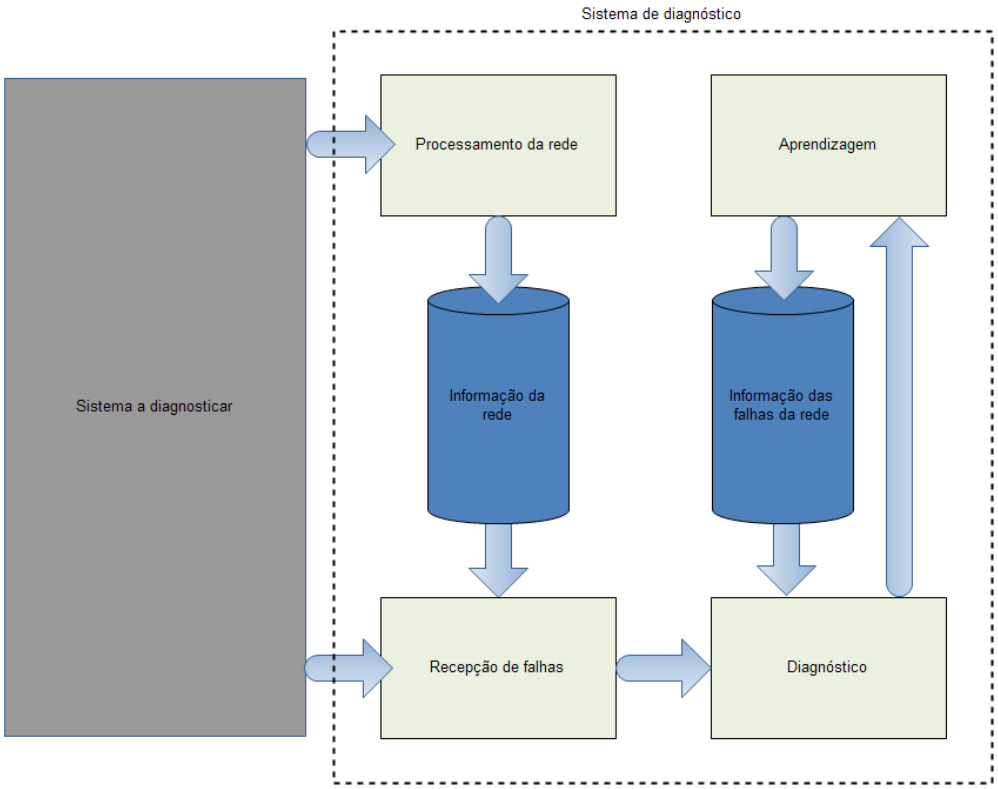

Figura 2 – Arquitectura do sistema de diagnóstico centralizado

O sistema de diagnóstico proposto é constituído por quatro módulos principais: *Processamento da rede*, *Recepção de falhas*, *Diagnóstico* e *Aprendizagem*. A informação dos dispositivos e o conhecimento sobre a propagação de falhas do sistema encontram-se nas estruturas: *Informação da rede* e *Informação das falhas da rede*, respectivamente.

Os objectivos e o funcionamento de cada módulo e a troca de informação serão explicados de seguida.

3.1.1. PROCESSAMENTO DA REDE

Cada dispositivo (físico ou lógico) do sistema que se pretende diagnosticar é representado no sistema de diagnóstico como um vértice de um grafo que representa a infra-estrutura onde o dispositivo se encontra. Cada dispositivo, ou componente, da rede é identificado pelo seu nome (único no universo de componentes), pelo tipo de dispositivo que representa e pelas interacções que tem com outros componentes.

O processamento da rede é responsável pelos processos necessários para a adição e remoção de componentes e interacções. Toda a informação do sistema sobre a rede de componentes é armazenada na estrutura *Informação da rede*.

Uma interacção representa uma ligação (Eléctrica, Mecânica, etc.) entre dois componentes e é interpretada pelo sistema como uma possível via de transmissão de falha. O grafo, constituído pelos vértices e pelas interacções (arcos) dos vários componentes, torna-se a base de suporte ao diagnóstico. Visto que, através dos componentes e respectivas interacções, torna-se possível formalizar e criar estruturas que representam todas as possíveis redes de falhas (propagações) entre os dispositivos da infra-estrutura.

Para além da gestão da informação dos componentes e das suas interacções, este módulo tem, ainda, como tarefa modelar toda a informação do sistema em estruturas que possam ser representadas e analisadas do ponto de vista de uma rede de vértices e arcos. Em cada uma das redes, um vértice representa um componente e um arco representa uma interacção (Eléctrica, Mecânica, etc.) entre dois componentes.

A Figura 3 demonstra um exemplo de uma infra-estrutura de dez componentes e respectivas interacções. O sistema de diagnóstico recebe esta informação, interpretando-a e formalizando-a do ponto de vista de uma rede de possíveis propagações de falhas entre os vários componentes. O módulo *Processamento da rede* cria as estruturas de dados necessárias para guardar a informação da rede, ficando esta disponível para ser utilizada pelo módulo *Recepção de falhas*.

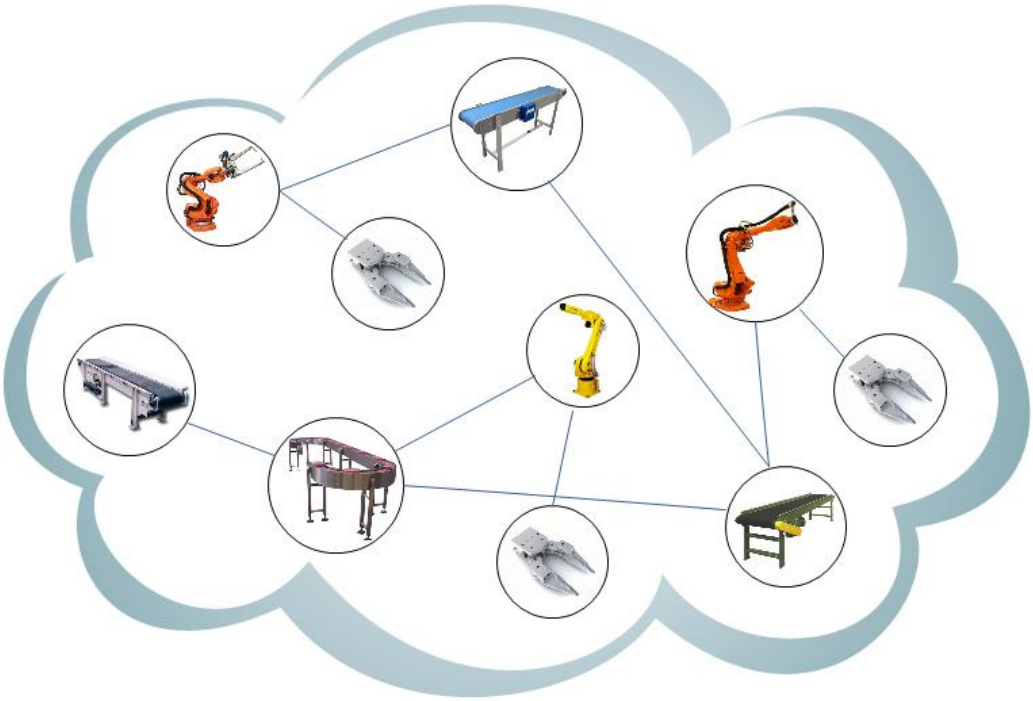

Figura 3 – Representação dos componentes como uma rede de falha

Como a infra-estrutura e respectivos componentes passam a ser representados como uma rede, torna-se apenas necessário fornecer a informação sobre o componente e as suas interacções para definir um novo elemento da rede.

Deste modo o conceito "*Plug-and-Play*" é aplicável uma vez que o sistema suporta a adição e remoção de qualquer componente independentemente do tipo de dispositivo representado por estes.

Para além dos dispositivos, toda a infra-estrutura em que os dispositivos estão inseridos torna-se transparente, o que permite a aplicação do sistema de diagnóstico em todos os tipos de infra-estruturas que possam ser representadas como uma rede de elementos com interacções entre si.

3.1.2. RECEPÇÃO DE FALHAS

Cada componente tem um sensor que detecta falhas de uma determinada natureza. Ao identificar uma falha, o dispositivo informa o sistema de diagnóstico e este recolhe e regista toda a informação considerada relevante para a análise da falha.

A recolha da informação deve-se ao facto do sistema proposto analisar uma falha num componente, tendo como base as observações de uma vizinhança. A utilização de uma vizinhança para analisar uma falha em detrimento da rede completa, torna-se necessária para manter o sistema escalável numa rede de grandes proporções.

A distância que define a vizinhança a ter em conta e o tempo que o sistema demora a recolher a informação sobre a mesma são configuráveis para cada componente. Esta distância representa, para o sistema de diagnóstico, o alcance que uma propagação de falha, originada num componente, pode atingir na rede do sistema.

A função do módulo de recepção de falhas é definir uma sub-rede de componentes para cada falha. Esta sub-rede contém a informação necessária para analisar essa falha e será utilizada pelo módulo de diagnóstico. Para a construção da sub-rede, o módulo acede à estrutura de dados *Informação da rede*, que contém a rede completa de componentes, assim como a distância considerada relevante para cada componente.

Para efeitos de diagnóstico o sistema proposto analisa e diagnostica as falhas e as possíveis propagações que sejam da mesma natureza. Assim, torna-se necessário filtrar as interacções pela sua natureza no processo de construção de cada sub-rede.

A Figura 4 ilustra um exemplo em que o *Conveyor* (assinalado com um circulo vermelho) entra em falha e a respectiva sub-rede é constituída por todos os seus componentes vizinhos que se encontram num raio igual a 2 (2 saltos de distância) e que estão identificados por circunferências de cor amarela.

Todos os componentes, inseridos na sub-rede (assinalados a amarelo), que acusem falha serão adicionados á informação sobre a falha do componente inicial para serem alvo de análise.

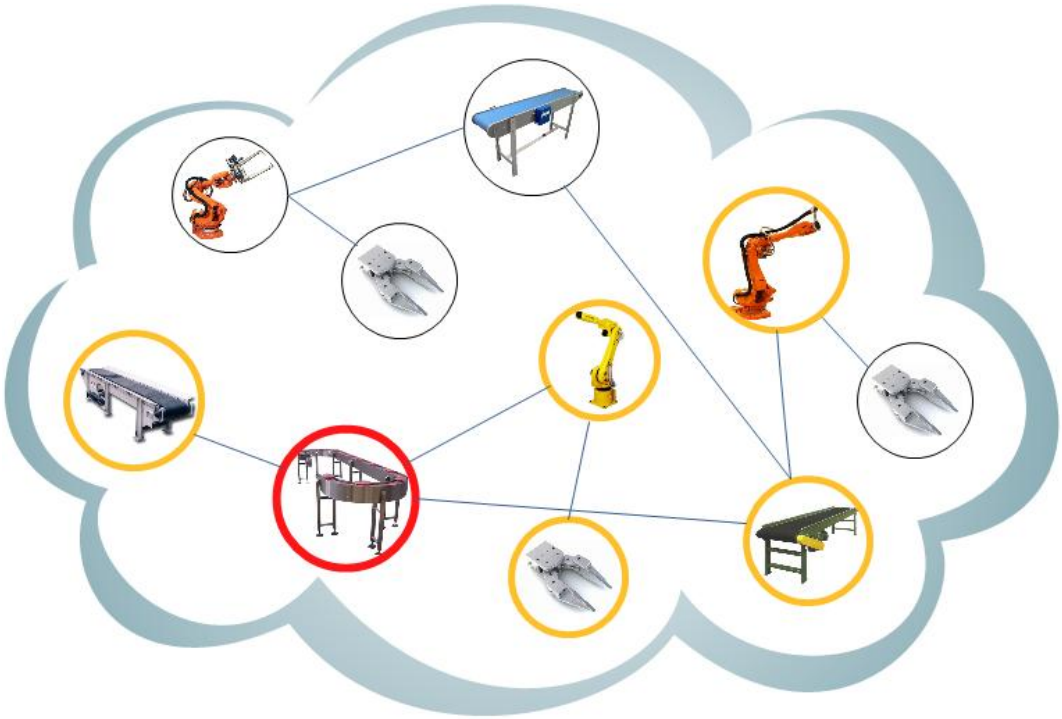

Figura 4 – Recepção de falha

Quando o tempo de recolha terminar considera-se reunida a informação que será alvo de análise e diagnóstico para explicar os fenómenos de falha presentes na sub-rede. Este tempo de recolha de informação, representa o tempo que o sistema aguarda para que uma falha se propague na rede para um componente vizinho, e é configurável, tal como a distância, para cada componente da rede. Assim, sempre que um novo componente detecta uma falha, e está dentro da área relevante à falha em questão, o sistema aguarda o tempo que estiver definido nesse mesmo componente.

3.1.3. DIAGNÓSTICO

Depois de reunida a informação sobre a falha, esta é enviada ao módulo de diagnóstico para ser processada. O processo de diagnóstico proposto utiliza o conhecimento adquirido através de falhas passadas para analisar e diagnosticar novas falhas. O conhecimento encontra-se armazenado na estrutura de dados *Informação das falhas da rede* (Figura 2) e representa para o sistema a informação aprendida sobre a propagação de uma falha entre dois tipos de componentes (*Robot*, *Conveyor*, etc.).

Na Figura 5 encontram-se dois exemplos que ilustram o tipo de informação que o sistema aprende e guarda sobre a propagação entre dois tipos de componentes.

No exemplo 5.a está representada a informação sobre a não existência de propagação de falha entre um componente do tipo *Robot* e um do tipo *Gripper*. No exemplo 5.b a informação armazenada indica que um componente do tipo *Robot*, em caso de falha, pode propagar essa falha a todos os seus componentes vizinhos que sejam do tipo *Conveyor*. Em ambas as imagens, as circunferências a verde e a vermelho, indicam a existência e a inexistência de falhas, respectivamente.

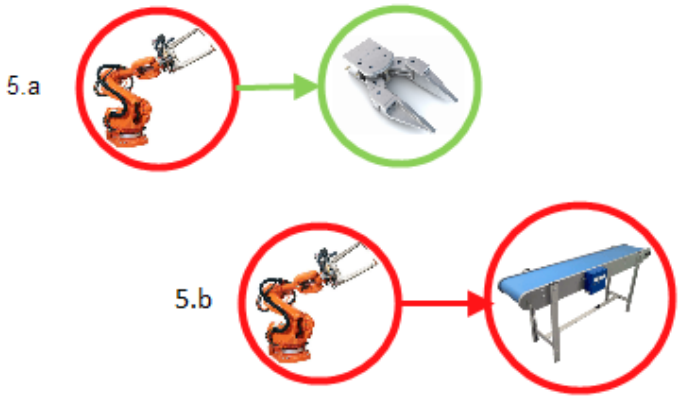

Figura 5 – Ilustração do tipo de informação sobre as falhas da rede
a) Caso em que não existe propagação de falha entre componentes
b) Caso em que existe propagação de falha entre componentes

A informação sobre a propagação de falha, entre dois tipos de componentes, pode ser conceptualmente representada pela seguinte expressão:

$$IR = (DTS, DTD, I, N) \quad (3)$$

Em que DTS e DTD representam o tipo de componente de origem e destino, respectivamente. O tipo de interacção entre a origem e destino é representado por I e N indica a existência ou inexistência de propagação de falha na respectiva interacção.

O módulo de diagnóstico analisa a informação da sub-rede aplicando o conhecimento que tem da informação das falhas da rede e diagnosticando a falha inicial e

a possível propagação desta. Para cumprir este objectivo, o módulo de diagnóstico deve ter em conta todos os componentes da sub-rede que sejam potenciais causadores da falha inicial. Considera-se como potencial causador da falha inicial qualquer componente da sub-rede que consiga alcançar o componente onde foi identificada a falha inicial.

Este processo de análise da informação está representado na Figura 6, onde são construídas redes de possíveis propagações de falhas através da utilização da informação da falha, recebida do módulo *Recepção de falhas*, e da informação que o sistema tem sobre falhas que ocorreram no passado. Todos os componentes identificados como potenciais causadores servem de input, individualmente, a este processo.

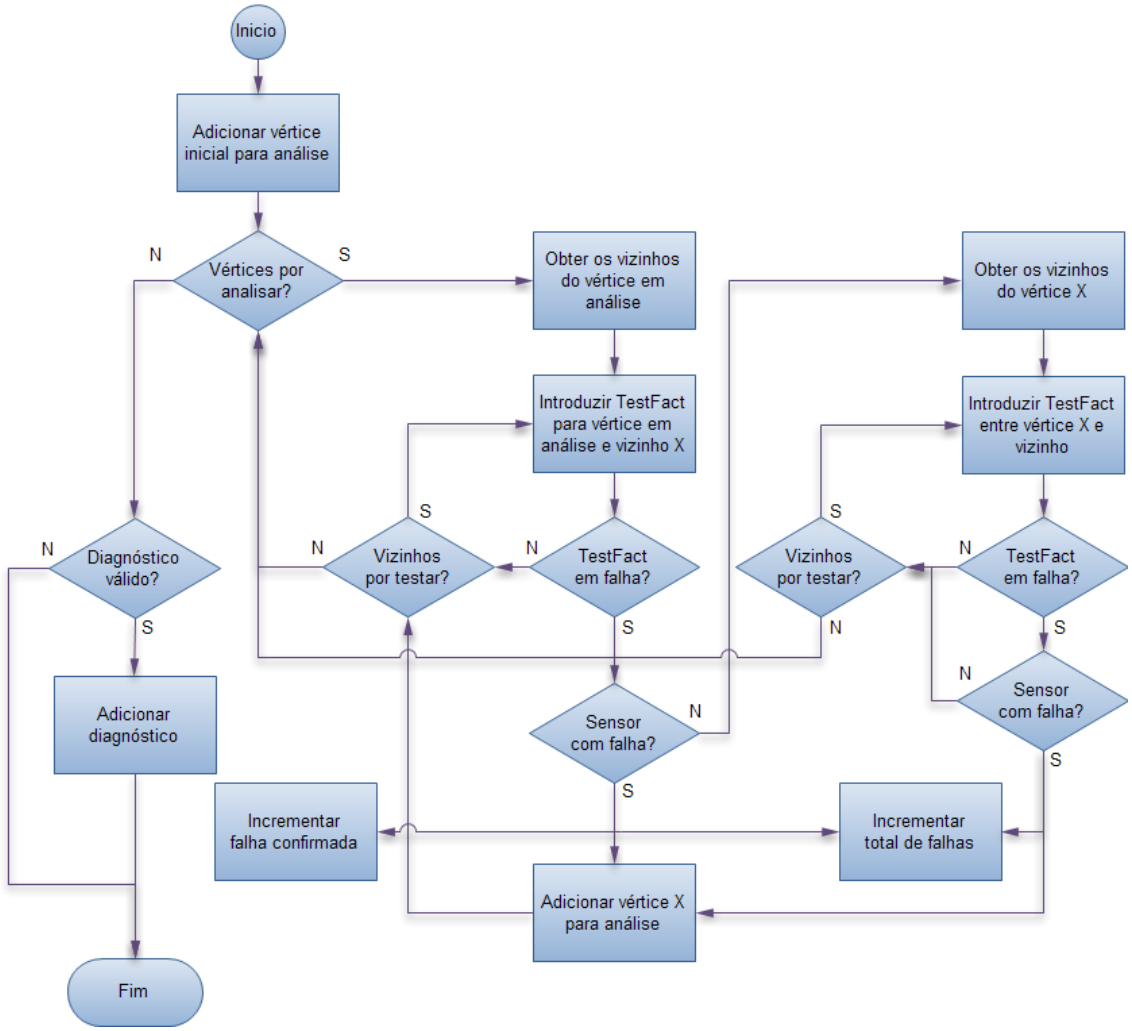

Figura 6 - Fluxograma de funcionamento do sistema de diagnóstico

Como diagnósticos válidos, consideram-se todos aqueles que têm o componente que desencadeou este processo (componente com a falha inicial) na sua rede de propagação de falha.

No caso da existência de vários diagnósticos plausíveis para a explicação da falha, é necessário que o módulo de diagnóstico tenha a capacidade de decidir um dos diagnósticos. Como factor de comparação entre os diagnósticos, e também como indicador da validade de um diagnóstico, definiu-se uma métrica de avaliação. Esta métrica é um indicador entre a informação verificada e não verificada sobre as propagações de falhas da rede, e é representada pela seguinte expressão:

$$score = \frac{\sum_0^c (value(IR_c))}{c} \quad (4)$$

Em que c é o numero total dos factos (com a informação de falhas), utilizados na elaboração do diagnóstico ("total de falhas" no diagrama), e $value(IR_c)$ é dado pela condição:

$$value(IR_{cn}) \begin{cases} 1, se\ IR\ é\ verificado \\ 0, caso\ contrário \end{cases} \quad (5)$$

Em que o somatório $\sum_0^c (value(IR_c))$ é igual ao contador "falha confirmado" representado no fluxograma da Figura 6.

Deste modo, o diagnóstico que reunir a avaliação mais alta será considerado como o que consegue explicar melhor a falha presente no sistema.

O sistema de diagnóstico proporciona ao utilizador a opção de introduzir manualmente um diagnóstico e alterar os diagnósticos devolvidos pelo método de diagnóstico. Para tal, é necessário o desenvolvimento de uma interface que permita ao utilizador visualizar a rede de componentes e interagir com esta, permitindo a identificação manual de falhas e a indicação da propagação destas através das interacções entre os componentes.

3.1.4. APRENDIZAGEM

Após a selecção, o diagnóstico é enviado ao módulo de aprendizagem com o objectivo de revalidar o conhecimento existente na estrutura *Informação das falhas da*

rede. Todas as falhas e propagações entre os componentes presentes no diagnóstico, são comparadas com a informação que o sistema possui das falhas da rede.

Ao realizar esta comparação, o módulo de aprendizagem chega a uma das seguintes situações:

- A informação existente no sistema sobre a propagação entre os dois tipos de componentes é confirmada;
- A informação existente no sistema sobre a propagação entre os dois tipos de componentes não é confirmada;
- Não existe nenhuma informação no sistema sobre a propagação entre os dois tipos de componentes.

A informação sobre a propagação entre dois tipos de componentes é válida enquanto não for alcançada uma determinada percentagem entre o número de vezes que foi confirmada e que não foi confirmada. Quando a percentagem é atingida a informação sobre a propagação é invertida, ou seja, se antes da comparação havia propagação entre os dois componentes depois da comparação o sistema assume que não há propagação entre os dois componentes.

Quando o resultado da comparação confirma a informação já existente no sistema, esta é reforçada ao ser incrementado o seu contador de informação confirmada. Se o resultado da comparação indicar que não existe nenhuma informação sobre a propagação entre dois componentes, essa informação é introduzida na estrutura *Informação sobre as fal*has da rede, sendo classificada como temporária.

Uma informação temporária sobre a propagação entre dois componentes implica que esta não seja usada para efectuar o diagnóstico. Quando a propagação da informação temporária é repetida um número de vezes, o módulo de aprendizagem retira a classificação temporária e a informação sobre a propagação entre os dois componentes passa a ser utilizada pelo módulo de diagnóstico.

Na Figura 7 observa-se um exemplo em que um *Conveyor* tem um vizinho *Robot* em falha e não existe nenhuma informação sobre a propagação entre eles. Nesta situação, o módulo de diagnóstico não consegue relacionar uma possível propagação entre estes dois componentes. No entanto, o módulo de aprendizagem, ao analisar o diagnóstico efectuado, introduz no bloco *Informação das falhas da rede* uma informação

temporária que representa a propagação (neste caso, a existência de propagação) entre os dois tipos de componentes.

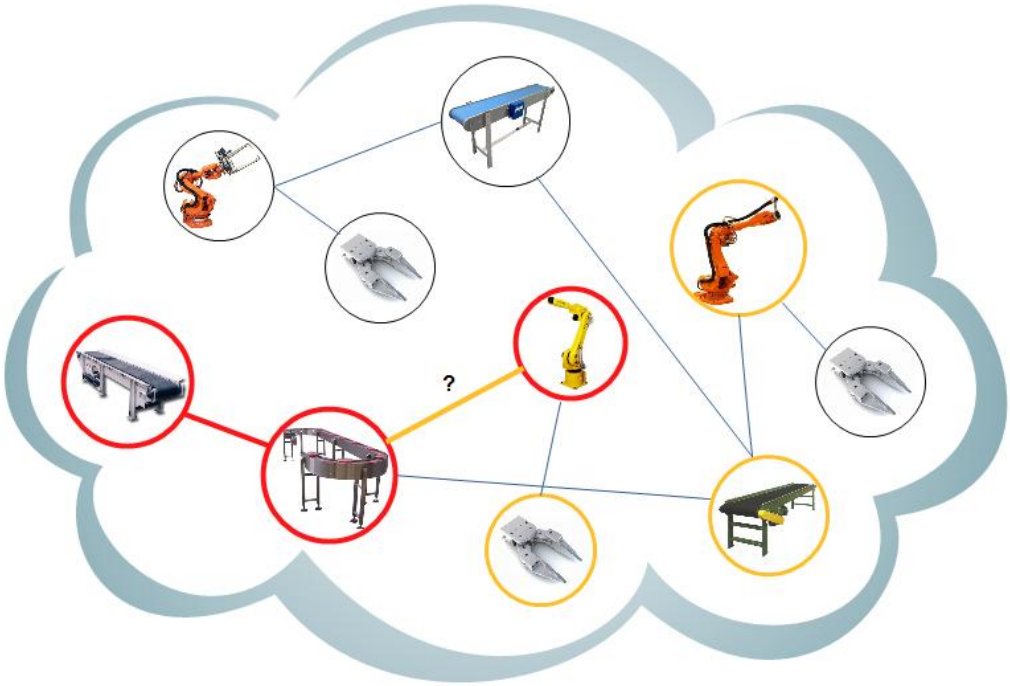

Figura 7 – Exemplo de aprendizagem

Este processo de aprendizagem é aplicado a todos os componentes que constituem a sub-rede e que estão presentes no diagnóstico escolhido pelo módulo de diagnóstico.

O resultado deste processo de aprendizagem traduz-se numa permanente actualização da informação extraída do sistema. Face a isto, espera-se que o sistema de diagnóstico centralizado apresente evolução e adaptação ao longo do seu funcionamento, e consiga reagir a alterações estruturais da rede.

3.2. DIAGNÓSTICO DISTRIBUÍDO

O sistema de diagnóstico distribuído, tem como objectivo realizar diagnóstico utilizando apenas a informação local disponível. Cada componente do sistema realiza o seu processo de diagnóstico e infere o seu estado, através do seu sensor de falha e através da análise e julgamento dos estados dos seus vizinhos directos.

Como tal, cada componente deve apenas ter acesso à informação dos seus vizinhos directos, e claro, à informação harmonizada do seu sensor. Deste modo, assegura-se a escalabilidade do sistema já que, independentemente do número de componentes do sistema, a informação que é necessária mantém-se constante.

Internamente, cada componente pode estar num dos seguintes estados:

- OK – O componente está a funcionar correctamente;
- NOK – O componente tem uma falha;
- PFO – Indica que o componente está em falha e está a propagar, essa mesma falha, para os seus vizinhos, através das suas interacções de saída;
- PFOther – O componente está afectado por uma falha com origem nos seus vizinhos, recebida através das suas interacções de entrada;
- PFOPFOther – Indica que o componente está a receber uma falha dos seus vizinhos, através das suas interacções de entrada, e está a propagar essa falha, através das interacções de saída, para os seus vizinhos.

A qualidade e o tipo de observações desempenham um papel fundamental na convergência do sistema e na eficiência do diagnóstico [65]. As observações que cada componente faz, através dos estados dos seus vizinhos, são definidas por cinco indicadores:

- Sensor – Estado do sensor de falha
- In Min – Minoria das interacções de entrada
- In Maj – Maioria das interacções de entrada
- Out Min – Minoria das interacções de saída
- Out Maj – Maioria das interacções de saída

Alguns dos símbolos das observações que cada componente faz, através dos cinco indicadores anteriormente referidos, encontram-se na Tabela 1.

Símbolos Observações	Sensor	In Min	In Maj	Out Min	Out Maj
Ok	0	0	0	0	0
IMaj	0	0	1	0	0
IMin	0	1	0	0	0
OF	1	0	0	0	0
OF_OMin	1	0	0	1	0
OF_IMaj	1	0	1	0	0
OF_IMaj_OMin	1	0	1	1	0
OF_IMin	1	1	0	0	0
OF_IMin_OMaj	1	1	0	0	1
OF_IMin_OMin	1	1	0	1	0

Tabela 1 – Lista abreviada dos estados definidos pelos vizinhos

Os valores 1 e 0 indicam a percepção, feita pelo componente, sobre as suas interacções e sobre o estado do seu sensor. É de salientar, que no total são 18 observações, já que algumas das combinações são impossíveis, tais como, minoria e maioria simultaneamente.

Um modelo probabilístico que utiliza o modelo escondido de Markov ("*Hidden Markov Model*" – HMM) é considerado para relacionar as observações e os estados de falha. Um HMM é definido por:

$$\lambda = (A, B, \pi) \quad (6)$$

onde A é uma matriz $N \times N$ (em que N é o numero de estados do modelo), que representa as probabilidades de transição de estados:

$$P(q_{t+1} = S_j | q_t = S_i) = a_{ij} \quad (7)$$

B é uma matriz $N \times M$ (em que Mé o numero de símbolos das observações), que inclui as probabilidades das observações:

$$P(O_t = k | q_t = S_i) = b_i(k) \quad (8)$$

(por exemplo, a probabilidade da observação k acontecer dado que o estado actual é S_i), e π representa as probabilidades iniciais dos estados.

Neste contexto, um HMM pode ser visto como uma máquina de estados finitos, cujas transições entre estados são probabilisticamente impulsionadas (a probabilidade de transição para outro estado dado o estado actual).

A representação de uma matriz com as probabilidades de transição de estado (A), encontra-se na Figura 8.

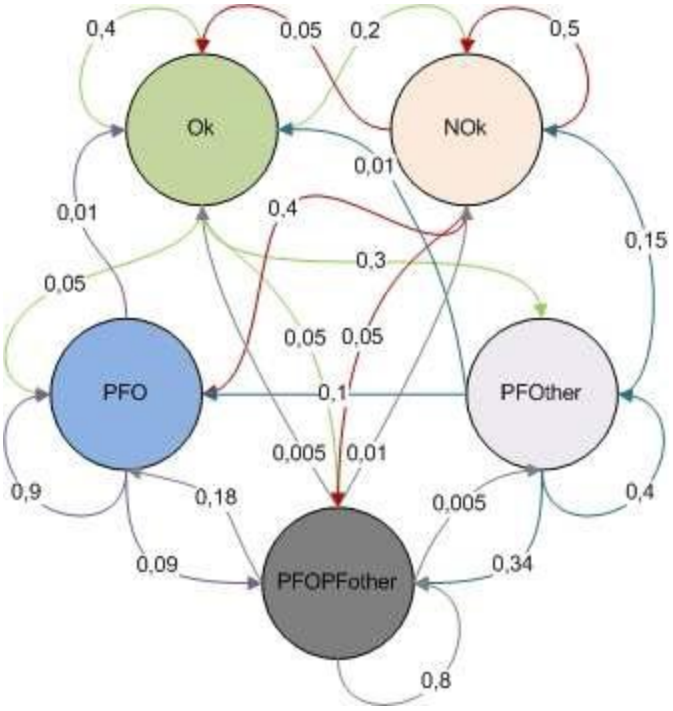

Figura 8 – Representação da matriz com a probabilidade de transição de estados

Como os componentes operam em conjunto, a informação operacional histórica pode ser usada. Cada componente pode assim processar a sua informação, ajustando o julgamento que faz sobre o seu estado interno, baseando-se naquilo que aprendeu através das interacções com os seus vizinhos.

Neste contexto, pode decidir que a informação que recolhe sobre os seus vizinhos é mais fiável do que a sua informação sensorial. Adicionalmente, se a informação sensorial não estiver disponível, continua a ser possível prever o seu estado.

3.3. DIFERENÇAS ENTRE ARQUITECTURAS

As duas arquitecturas dos sistemas de diagnóstico, anteriormente apresentadas, tem objectivos em comum mas diferem em alguns aspectos importantes.

Nas características comuns pode-se destacar a capacidade de suportar alterações na rede (adição e remoção de componentes e interacções), e a capacidade de identificar o estado de cada componente perante uma falha. Ambos os processos possuem a capacidade de aprendizagem ao longo do funcionamento do sistema.

No entanto, as arquitecturas diferem em algumas características importantes, como o tipo de informação utilizado e o método de raciocínio empregue para realizar diagnóstico.

O sistema centralizado tem ao seu dispor toda a informação existente, enquanto no distribuído apenas a informação local está disponível. Isto faz com que a capacidade de processamento no centralizado seja elevada e dependa do tamanho da rede, ao invés do sistema distribuído, que é imune ao tamanho da rede já que está embutido no próprio componente.

O método de diagnóstico, utilizado pelas duas arquitecturas, é uma das principais características dos sistemas. No primeiro caso é utilizado raciocínio lógico e temporal, apresentando limitações ao nível do tempo de propagação de falha, já que o processo de reunir a informação, para realizar o diagnóstico, está temporalmente limitado. O sistema distribuído utiliza um modelo probabilístico (HMM) e é assíncrono, isto é, o processo de diagnóstico reage “on the fly” a alterações na rede.

O tipo de propagação das falhas é indiferente para o sistema distribuído, ou seja, não depende do tipo de componentes envolvidos na falha, mas sim do estado de cada um para utilizar o HMM. Já o sistema centralizado tenta identificar padrões de falhas na rede, através do conhecimento que tem sobre a rede, e como tal o tipo de propagação existente (tipo de componentes em falha) influencia o seu desempenho.

Na Tabela 2 encontram-se, lado a lado, as principais características das arquitecturas dos dois sistemas de diagnóstico em estudo.

	Sistema centralizado	Sistema distribuido
Dimensão do conhecimento	Global	Local
Método de diagnóstico utilizado	Raciocínio lógico e temporal	Probabilístico
Suporte de alterações na rede de componentes	Sim	Sim
Atraso na propagação da falha (delay)	Limitado às características do sistema	Assíncrono
Tipo de propagação preferencial	Padrões de falhas	Indiferente
Capacidade de aprendizagem	Sim	Sim
Identificação do estado de falha de cada componente	Sim	Sim
Identificação do caminho de propagação da falha	Sim	Não
Capacidade de processamento	Elevada	Baixa

Tabela 2 – Principais características das arquitecturas

A comparação dos sistemas de diagnóstico baseia-se na identificação do estado (de falha) de cada um dos componentes da rede. Isto é, a identificação correcta do componente, ou componentes, que estão na origem da falha e na identificação dos componentes que estão a ser afectados, ou a afectar outros, devido á propagação de falhas.

Para se proceder à comparação é necessário providenciar, a ambos os sistemas de diagnóstico, condições de teste válidas e adequadas, isto é, a criação de várias redes de componentes com diferentes níveis de complexidade. Deste modo, e através destas redes, torna-se possível avaliar correctamente o desempenho de cada sistema de diagnóstico perante diferentes graus de complexidade.

4. IMPLEMENTAÇÃO

Este capítulo descreve a implementação detalhada da arquitectura do sistema centralizado proposto no capítulo 3.

4.1. DIAGNÓSTICO CENTRALIZADO

A implementação da arquitectura proposta foi feita em quatro etapas. Cada uma corresponde a um módulo: processamento da rede, recepção de falhas, diagnóstico de falhas e aprendizagem resultante dos diagnósticos. A implementação de cada etapa será descrita neste subcapítulo assim como as ferramentas que foram utilizadas para tal. A interface da aplicação e suas funcionalidades serão apresentadas e explicadas na aquisição da topologia da rede e no diagnóstico de falhas visto a interface estar directamente relacionada com estes módulos.

4.1.1. BIBLIOTECAS DE SOFTWARE UTILIZADAS

O sistema de diagnóstico centralizado foi desenvolvido sobre a plataforma de desenvolvimento de agentes *JADE* ("*Java Agent Development Framework*") [66], que fornece todos os mecanismos necessários de comunicação (*FIPA Request interaction Protocol*), sincronização e interoperabilidade.

Para além do *JADE* utilizou-se a biblioteca de software *JUNG* ("*Java Universal Network/Graph Framework*") [67] e a plataforma *Drools* [68], mais especificamente, o *Drools Expert* e o *Drools Fusion*.

4.1.1.1. JUNG

A biblioteca de software *JUNG* permite visualizar, analisar e modelar informação que pode ser visualizada como um grafo ou uma rede. Providencia vários meios de pesquisa e organização de informação (caminhos entre nós, caminho mais curto,

distancia entre nós, etc.) que se tornam extremamente úteis para o processamento de informação num sistema de diagnóstico que tem a representação da informação em rede.

A Figura 9 mostra um exemplo de uma rede visualizada pelo *JUNG*. A rede é constituída por cinco vértices e sete arcos (interacções) direccionais.

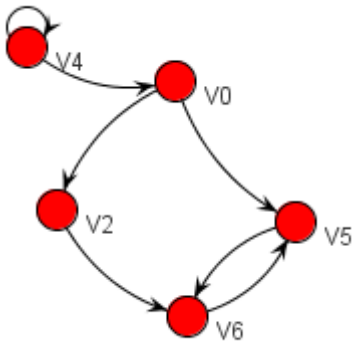

Figura 9 – Exemplo de rede no *JUNG*

Os processos de adição, alteração e remoção de elementos da rede são feitos através de funções simples, pré-definidas pelo *JUNG*. A rede presente na Figura 9 é um grafo directo mas existem outros tipos de grafos diferentes no JUNG, cada um com as suas funcionalidades.

Esta biblioteca suporta ainda por defeito mecanismos que providenciam uma interface directa com o utilizador permitindo acrescentar, editar, remover e mover visualmente (zoom e transformação gráfica) a informação relacionada com a rede de uma forma simples, eficaz e directa.

4.1.1.2. Drools

O *Drools* é um software de gestão de decisões lógicas que utiliza um motor de inferência, tendo em conta regras e factos. Uma regra representa uma determinada condição a que os factos são submetidos e os factos podem ser vistos como o conhecimento a testar. As regras são inicialmente carregadas para o *Drools* e, posteriormente, os factos ao serem introduzidos são automaticamente relacionados com

as regras. Este sistema permite desenvolver, manter, monitorizar, testar, organizar e criar novas regras e factos em variadíssimos contextos, suportando a integração de diferentes linguagens e diversas configurações.

O *Drools* é constituído por quatro módulos diferentes, cada um com determinadas funções. No sistema de diagnóstico centralizado utilizou-se o *Expert* e o *Fusion* que fornecem um motor de inferência (raciocínio lógico) e uma sequencia temporal para relacionar informação.

- Drools Expert

O *Drools Expert* representa o próprio motor de inferência que permite criar, manter, remover e relacionar factos e regras. Pode ser visto como quatro blocos distintos: a memória que contém as regras (*Production Memory*), a memória que contém os factos (*Working Memory*), o algoritmo de combinação das regras e dos factos (*Pattern Matcher*) e, por último, o bloco responsável por armazenar e processar os resultados (*Agenda*).

A Figura 10 representa a interacção entre os vários blocos do *Expert*.

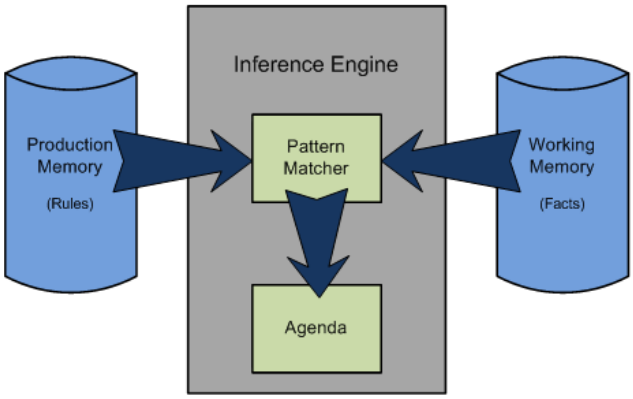

Figura 10 – *Drools Expert*
(Imagem adaptada de [69])

Um exemplo do funcionamento do *Drools* será explicado de seguida. Na Figura 11 estão representadas quatro estruturas que serão inseridas na *Working Memory* (como factos).

```
public class Room {
    String name;
}

public class Sprinkler {
    Room room;
    boolean on;
}

public class Fire {
    Room room;
}

public class Alarm {
}
```

Figura 11 – Exemplo de factos

As regras estão contidas num ficheiro com a extensão *drl* que é carregado para *Production Memory*, na Figura 12 estão presentes três regra que interagem com os factos inseridos na *Working Memory*.

```
rule "When there is a fire turn on the sprinkler"
when
    Fire($room : room)
    $sprinkler : Sprinkler( room == $room, on == false )
then
    modify( $sprinkler ) { setOn( true ) };
    System.out.println("Turn on the sprinkler for room" + $room.getName());
end

rule "Raise the alarm when we have one or more fires"
when
    exists Fire()
then
    insert( new Alarm() );
    System.out.println( "Raise the alarm" );
end

rule "Cancel the alarm when all the fires have gone"
when
    not Fire()
    $alarm : Alarm()
then
    retract( $alarm );
    System.out.println( "Cancel the alarm" );
end
```

Figura 12 – Exemplo de regras

A primeira regra verifica a existência de fogos nos quartos e o estado dos "extintores" do respectivo quarto. No caso de estarem desligados, os "extintores" são ligados. A regra é automaticamente testada sempre que é introduzido um fogo ou um "extintor" na *Working Memory*. Nas regras seguintes o princípio de funcionamento é o mesmo. Apenas muda o resultado da regra, na segunda regra quando existe um fogo é inserido um alarme na *Working Memory* e na terceira regra os alarmes são removidos da *Working Memory* se não houver nenhum fogo.

- Drools Fusion

O *Fusion* acrescenta ao *Drools* a opção de classificar os factos como eventos, o que introduz a componente de tempo. Não representa o tempo real dos factos mas sim a noção cronológica entre vários factos, o que torna possível definir sequências de eventos. Devido à componente temporal, consegue-se induzir atrasos no processamento das regras, ou seja, torna-se possível obrigar o sistema a aguardar um tempo pré-definido até que uma determinada regra seja testada.

Uma mais-valia de usar o *Fusion* é a capacidade do motor de inferência perceber quando é que um facto (classificado como evento) já não é necessário para testar as regras existentes. Através das relações temporais nas regras, os factos passam a ser dinâmicos e a gestão da memória (*Working Memory*) é automaticamente feita pelo próprio *Drools* deixando de haver a preocupação e a necessidade da remoção manual dos factos.

4.1.2. PROCESSAMENTO DA REDE

Cada componente da rede, implementado sobre a plataforma *JADE*, regista-se no sistema de diagnóstico através do envio de uma mensagem com o seu nome, o tipo de dispositivo que representa e o tipo de interacções que suporta. Posteriormente, são trocadas mensagens com as interacções que o agente tem com os outros. Esta informação é dinâmica, ou seja, é possível alterar (adicionar e remover) as interacções entre os agentes.

Todas as trocas de mensagens obedecem ao protocolo *FIPA Request* (Figura 13).

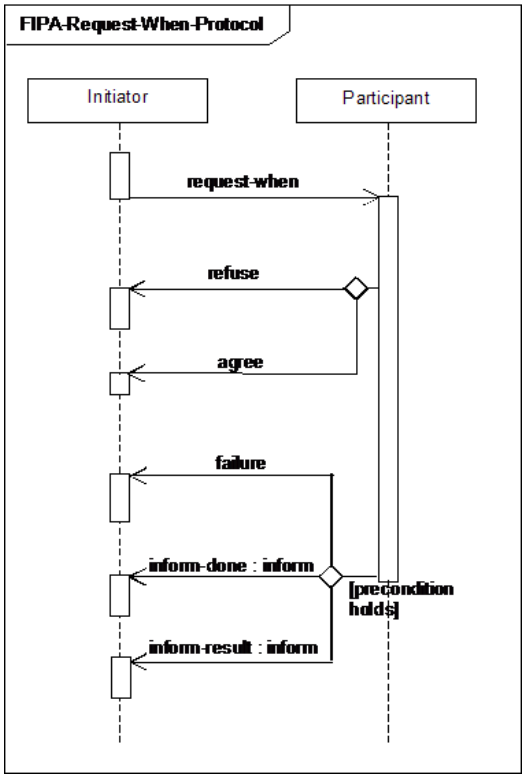

Figura 13 – Protocolo *FIPA Request*
(Imagem retirada de [70])

A Figura 14 descreve todos os casos de uso (relacionados com a topologia da rede) entre um agente da rede e o sistema de diagnóstico, registo e remoção da rede e adição e remoção de interacções entre agentes.

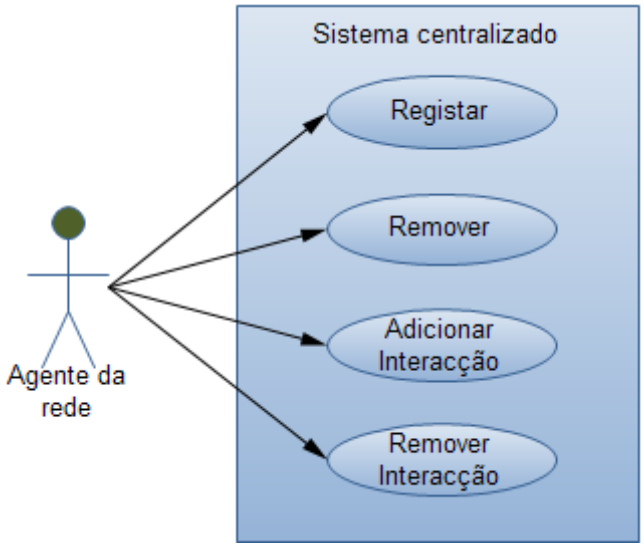

Figura 14 – Casos de uso entre um componente da rede e o sistema centralizado

Nas Figuras 15 e 16 estão representados os diagramas de sequência da troca de mensagens entre cada agente da rede e o sistema de diagnóstico para o registar e remover, respectivamente. Tanto no registo como na remoção, o envio de cada uma destas mensagens, implica a actualização de todos os outros componentes da rede que tem interacções com componente (representado pelo agente) que enviou a mensagem.

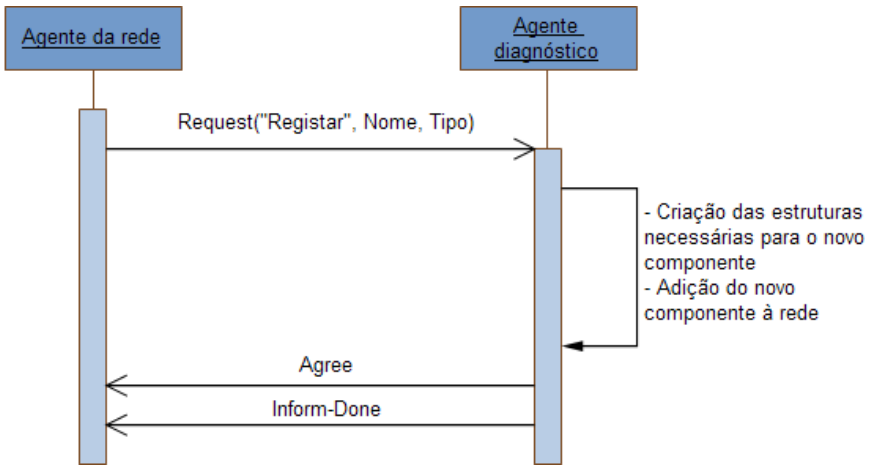

Figura 15 – Diagrama de sequência para o registo de um agente

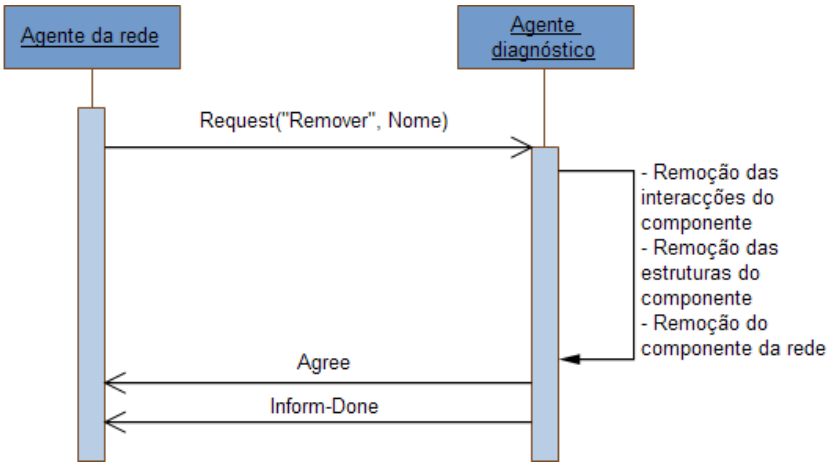

Figura 16 – Diagrama de sequência para a remoção de um agente

Nas Figuras 17 e 18 encontram-se os diagramas da sequência de mensagens trocadas no caso da adição e remoção de interacções entre componentes. Analogamente às trocas de mensagens para o registo e remoção de componentes, também a adição e remoção de interacções implica uma actualização das estruturas dos componentes nos quais se pretende estabelecer uma interacção.

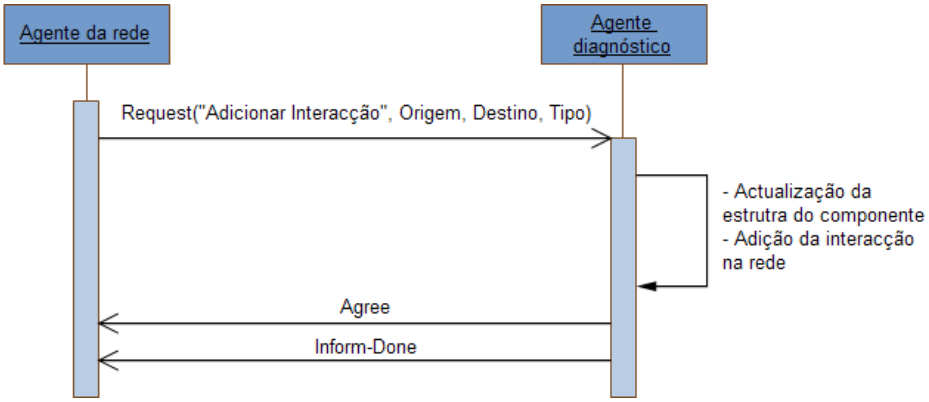

Figura 17 – Diagrama de sequência para a adição de uma interacção entre dois agentes

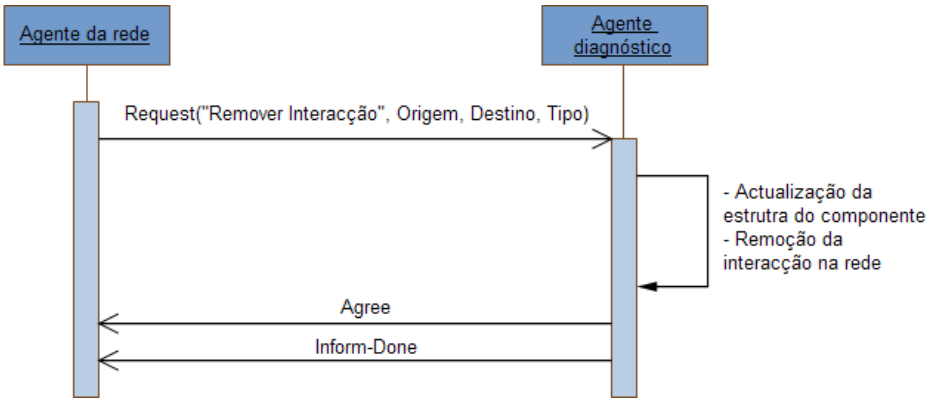

Figura 18 – Diagrama de sequência para a remoção de uma interacção entre dois agentes

Com o objectivo de ter todos os componentes da rede representados como vértices e as suas interacções como arcos, definiu-se a estrutura *Node* (Tabela 3), para guardar a informação de cada componente (denominado como vértice daqui em diante) e os tipos de interacções.

Node	
Name	Contém o nome do vértice (componente)
Type	Contém o tipo de dispositivo que o vértice representa (*Conveyor*, *Pallet*, *Gripper*, etc)
Connection	Define o tipo de interacção entre este vértice e os vértices seguintes e anteriores
PreviousNodes	Contém todos os vértices com os quais este tem uma interacção de saída
NextNodes	Contém todos os vértices com os quais este tem uma interacção de entrada

Tabela 3 – Estrutura *Node*

Um conjunto de estruturas *Node* providencia a informação necessária para criar várias redes consoante os tipos de interacções existentes. Segue-se um exemplo de três vértices (*Conveyor 1*, *Conveyor 2* e *Robot 1*) que, entre si, formam duas redes distintas (Eléctrica e Fluxo).

Node	
Name	Conveyor 1
Type	Conveyor
Connection	Eléctrica

NextNodes	Conveyor 2

Tabela 4 – Informação Eléctrica do *Conveyor 1*

Node	
Name	Conveyor 1
Type	Conveyor
Connection	Fluxo
PreviousNodes	Robot 1

Tabela 5 – Informação de Fluxo do *Conveyor 1*

Node	
Name	Conveyor 2
Type	Conveyor
Connection	Eléctrica
PreviousNodes	Conveyor 1

Tabela 6 – Informação Eléctrica do *Conveyor 2*

Node	
Name	Robot 1
Type	Robot
Connection	Fluxo
NextNodes	Conveyor 1

Tabela 7 – Informação de Fluxo do *Robot 1*

As quatro estruturas representadas em cima são agrupadas tendo em conta os tipos de interacções, resultando em dois conjuntos de estruturas que representam duas redes de interacções: uma rede com as interacções Eléctricas e outra com as interacções de Fluxo.

De modo a fornecer uma representação gráfica das várias redes que formalizam os componentes e as suas interacções, assim como os resultados do método de diagnóstico, foi desenvolvida uma interface que se encontra representada na Figura 19. Uma descrição das suas funcionalidades, será feita de seguida.

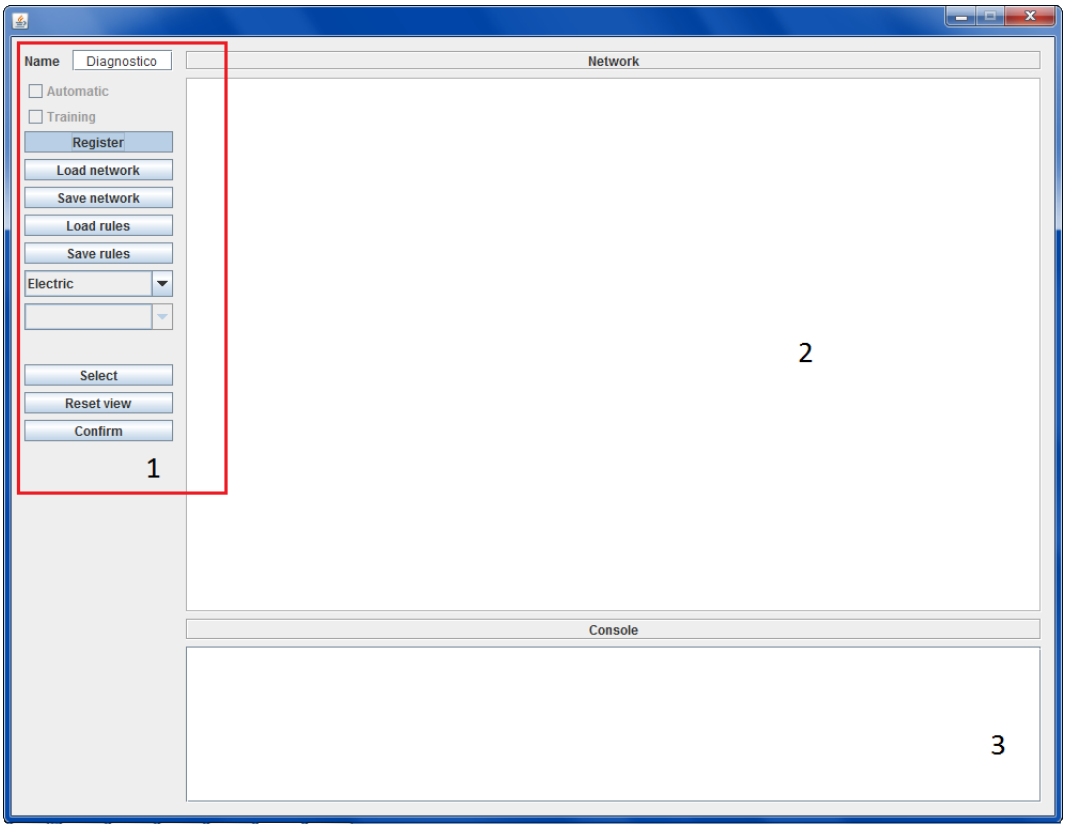

Figura 19 – Interface do sistema centralizado

As áreas 2 e 3, assinaladas na Figura 19, indicam a área de visualização da rede e a consola da aplicação, respectivamente. A Figura 20 revela as funcionalidades assinaladas pela área 1 da Figura 19 e na Tabela 8 descreve-se cada uma dessas funcionalidades.

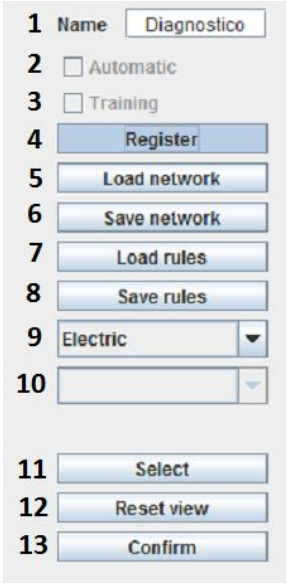

Figura 20 – Funcionalidades da interface

Funcionalidades da interface	
1	Nome do sistema centralizado (agente)
2	Opção para colocar o diagnóstico em modo automático (útil para testes)
3	Opção para colocar o sistema de diagnóstico em modo de aprendizagem
4	Registo do sistema de diagnóstico
5	Botão para carregar uma rede previamente guardada
6	Botão para salvar uma rede
7	Botão para carregar regras (factos) para o sistema
8	Botão para salvar as regras (factos) existentes no sistema
9	Escolha da rede (interacção) visualizada na área 2
10	Opção de escolha dos vários diagnósticos (quando as opções 2 e 3 não estão seleccionadas)
11	Opção para interagir graficamente com a rede visualizada da área 2
12	Botão para anular os efeitos visuais através da opção 11
13	Botão para confirmar o diagnóstico apresentado (quando as opções 2 e 3 não estão seleccionadas)

Tabela 8 – Descrição das funcionalidades da interface

Para visualizar a rede de componentes e as suas interacções na interface, e de certa forma trabalhar com a informação da rede, utilizou-se a biblioteca *JUNG*. A utilização desta biblioteca, na implementação da interface, será explicada de seguida.

Utilizou-se a biblioteca JUNG para criar as várias redes de vértices e arcos, que depois podem ser modulados em vários aspectos como visualizar apenas determinados vértices e arcos, mudar a sua forma, a cor, o tamanho, etc. O *JUNG* permite uma interacção directa entre o utilizador da aplicação e a informação visualizada na rede, mover vértices e arcos, fazer *zoom* à rede, seleccionar um ou mais vértices (e também arcos), entre outros. Esta funcionalidade de interacção do Jung, foi implementada na interface para tornar possível, ao utilizador desta, reajustar facilmente a visualização gráfica da rede. Todos estes mecanismos são praticamente implementados automaticamente pelo JUNG sem ser necessário recorrer a qualquer tipo de programação.

Para visualizar a rede foi necessário, primeiro, criar uma variável (denominado *graph*) de um determinado tipo de grafo. O *graph* contém a rede (vértices e arcos) e permite a sua configuração. Face ao objectivo do problema, optou-se por escolher um grafo directo, isto é, um grafo onde os vértices estão ligados entre si com arcos que têm sentidos. A Figura 21 exemplifica um grafo directo.

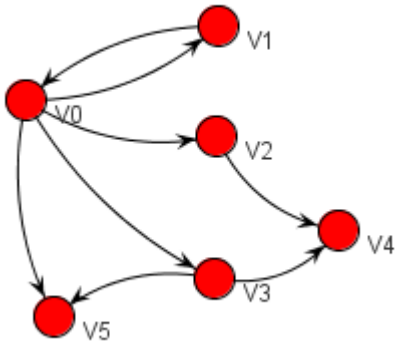

Figura 21 – Exemplo de um grafo directo

Como o nome de cada vértice é único, isso faz com que um arco entre dois vértices possa ser exclusivamente identificado com uma concatenação do nome dos seus vértices e do tipo de interacção. Através da Figura 21 e assumindo que o tipo de interacção da rede é de Fluxo o arco que sai do vértice *V0* para o vértice *V1* pode ser identificado exclusivamente entre todos os outros arcos como *VOV1-Fluxe,* assim como os respectivos vértices por *VO* e *V1*.

A configuração do *graph* inclui a forma dos vértices e dos arcos, a cor (interior e do bordo) que estes devem tomar consoante as várias situações, o tamanho, a legenda (para cada vértice e arco) e a própria visualização, já que é possível escolher quais os vértices e nós que aparecem ou não. A forma escolhida foi a circular para os vértices e a linha curva para os arcos, sendo a legenda de cada um o seu nome. Com os vértices e os arcos a serem identificados por uma *String* definiu-se no *graph* que ambos seriam deste tipo. Como tal, para adicionar ou remover vértices ou arcos basta aceder ao *graph* e usar as funções *AddVertex(String vertex)*, *RemoveVertex(String vertex)*, *AddEdge(String edge)* e *RemoveEdge(String edge)*. Para guardar o estado e realizar a manipulação da cor dos vértices e dos arcos foram criadas duas *HashMap* (*VertexMap* e *EdgeMap*) sendo os nomes dos vértices e dos arcos as respectivas chaves.

A estrutura de cada elemento (*Vertex*) do *VertexMap* encontra-se na Tabela 9 e a estrutura de cada elemento (*Edge*) do *EdgeMap* encontra-se na Tabela 10.

Vertex	
Name	Nome do vértice
SensorStatus	Estado que o sensor do vértice (agente) indica
DiagnosticStatus	Estado que o sistema de diagnóstico indica

Tabela 9 – Estrutura *Vertex*

Edge	
Name	Nome do arco
DiagnosticStatus	Estado que o sistema de diagnóstico indica

Tabela 10 – Estrutura *Edge*

Na configuração do *graph* a cor de cada vértice e de cada arco depende dos estados indicados pelo elemento correspondente no *VertexMap* e no *EdgeMap*. Os possíveis estados, significados e respectivas cores encontram-se nas Tabelas 11, 12 e 13.

Estados do sensor dos vértices		
Ok	Verde	O sensor do vértice não identifica nenhuma falha
NOk	Laranja	O sensor do vértice identifica uma falha

Tabela 11 – Estados do sensor dos vértices

Estados do diagnóstico dos vértices		
D_Ok	Azul	O diagnóstico indica que o vértice não tem falha e confirma o estado Ok do sensor
D_NOk	Vermelho	O diagnóstico indica que o vértice está em falha e confirma o estado NOk do sensor
D_PNOk	Magenta	O diagnóstico indica que o vértice está em falha e contraria o estado Ok do sensor
D_SNOk	Preto	O diagnóstico indica que o vértice é a origem da falha e confirma o estado NOk do sensor
D_SPNOK	Cinzento	O diagnóstico indica que o vértice é a origem da falha e contraria o estado Ok do sensor

Tabela 12 – Estados do diagnóstico dos vértices

Estados do diagnóstico dos arcos		
D_Ok	Verde	O diagnóstico indica que não há nenhuma falha a ser propagada entre os vértices
D_NOk	Vermelho	O diagnóstico indica que há uma falha a ser propagada entre os vértices

Tabela 13 – Estados do diagnóstico dos arcos

A configuração do *graph* é feita através de funções pré-definidas pelo *JUNG*, onde é apenas necessário programar o que cada função retorna, como por exemplo, a cor desejada do vértice. Quando os estados dos vértices e arcos são modificados, a cor dos mesmos é alterada consoante o estado.

Os exemplos das Tabelas 4, 5, 6 e 7, encontram-se representados para a rede Eléctrica na Figura 22 e para a rede de Fluxo na Figura 23. Aos exemplos referidos, acrescentou-se algumas situações de diagnóstico que ilustram os estados referidos na Tabelas 11, 12 e 13.

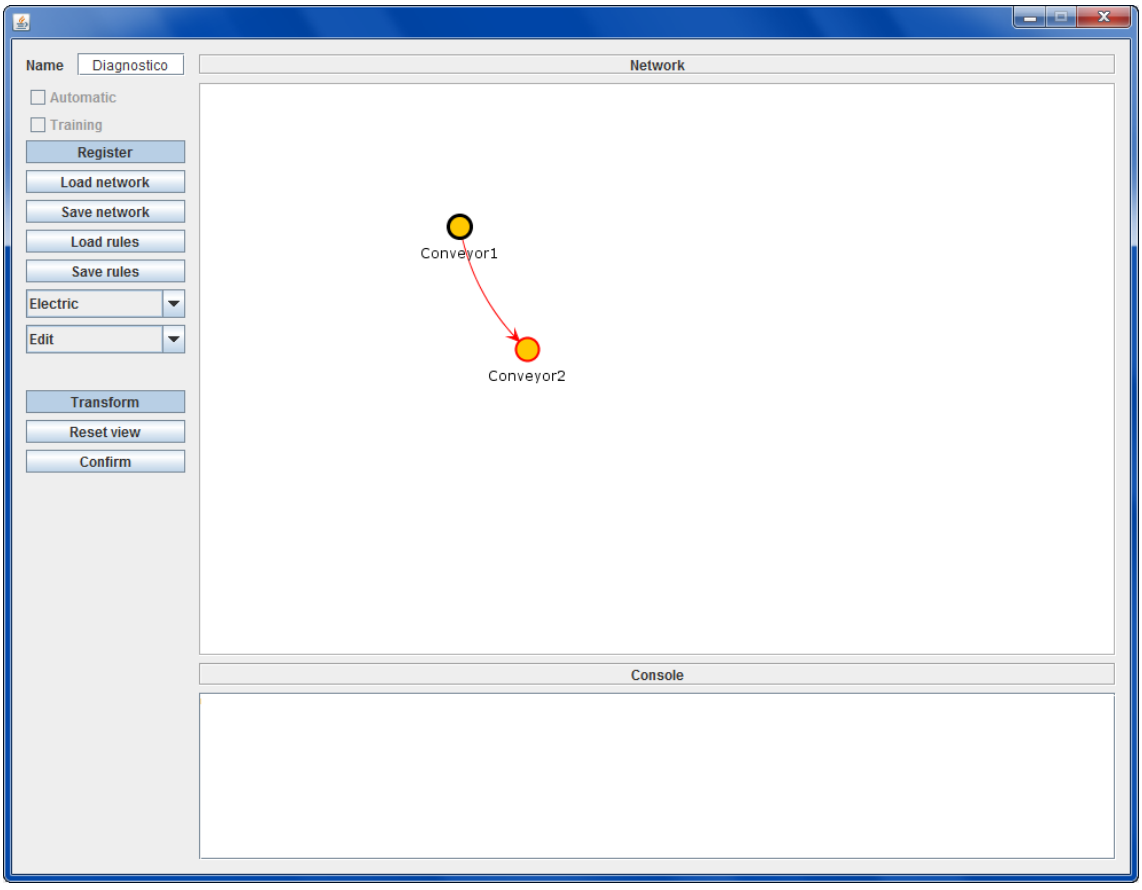

Figura 22 – Exemplo do diagnóstico na rede Eléctrica

Na Figura 22, ambos os sensores do *Conveyor1* e do *Conveyor2* assinalam falha. O sistema de diagnóstico indica que o *Conveyor1* está a causar a falha e a propagá-la ao *Conveyor2*.

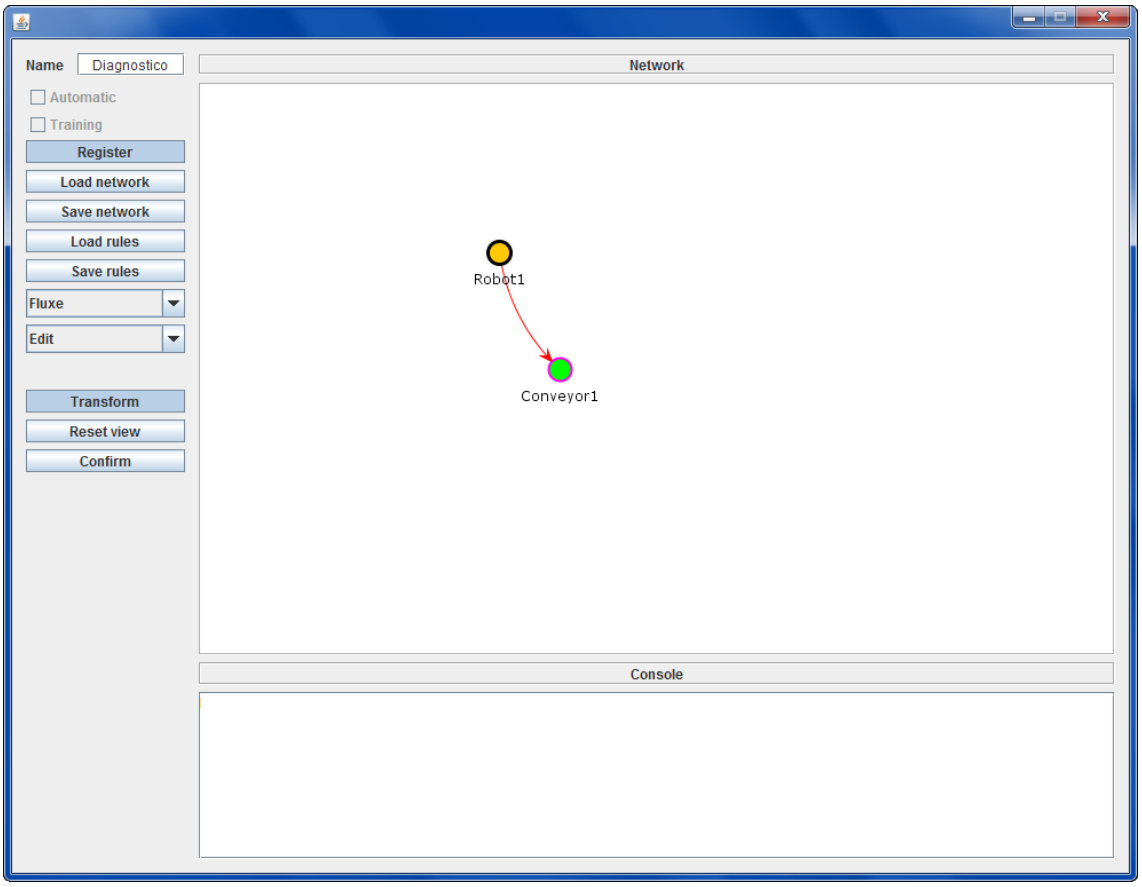

Figura 23 – Exemplo do diagnóstico na rede de Fluxo

Na Figura 23, apenas o sensor do *Robot1* assinala falha, mas o diagnóstico indica que o *Conveyor1* também está em falha, devido à falha originada no *Robot1*.

4.1.3. RECEPÇÃO DE FALHAS

Cada dispositivo tem um sensor de falha para cada um dos tipos de interacções que suporta. Quando uma falha é detectada, é enviada uma mensagem ao sistema de diagnóstico com a informação sobre o tipo de falha. A mensagem enviada respeita o protocolo *FIPA Request* (Figura 13) e aos casos de uso representados na Figura 14 acrescenta-se, agora, o envio da informação de falha completando os casos de uso entre os dispositivos da rede e o sistema de diagnóstico (Figura 24).

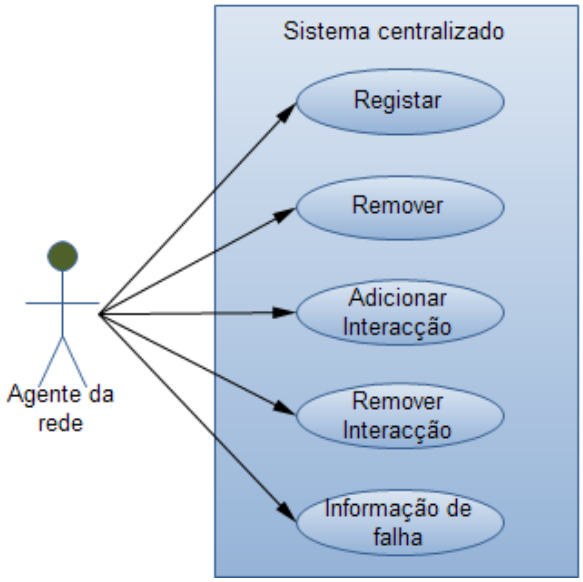

Figura 24 - Casos de uso entre um dispositivo da rede e o sistema centralizado

Na Figura 25, encontra-se o diagrama de sequência da troca de mensagens relacionadas com a informação do sensor entre cada agente da rede e o sistema de diagnóstico. Quando um dos componentes detecta uma falha, através do seu sensor, é enviada uma mensagem que desencadeia, no bloco *Recepção de falhas*, o processo de reunir a informação sobre a falha, que mais tarde será utilizada no processo de diagnóstico.

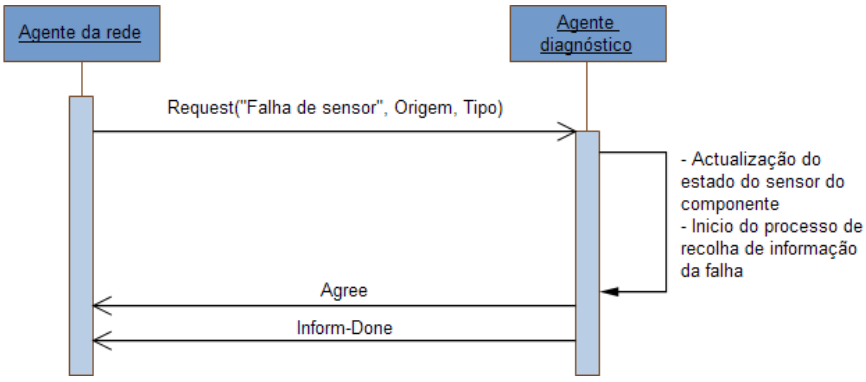

Figura 25 – Diagrama de sequência para indicar uma falha no sensor

Ao receber uma mensagem de falha, o sistema de diagnóstico verifica através do *JUNG* quais são os vértices que se encontram à distância pré-definida.

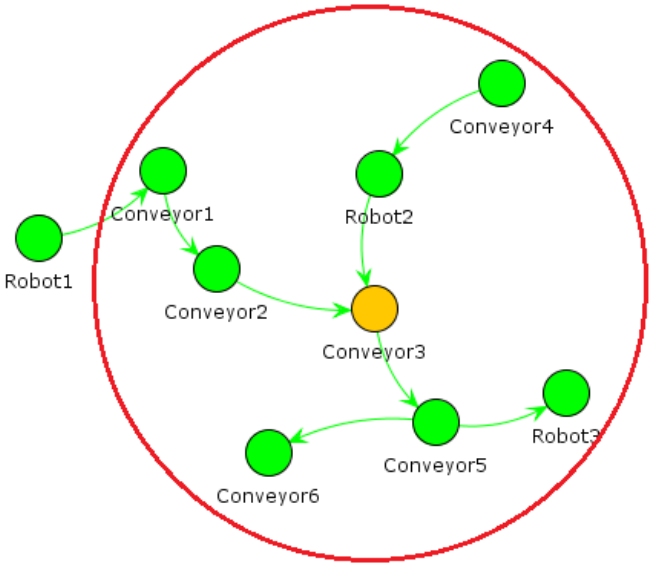

Figura 26 – Exemplo de falha no Conveyor3 com distância 2

Na Figura 26 pode-se verificar um exemplo de uma rede do tipo Eléctrica, em que a distância definida para o vértice *Conveyor3* é 2. Como tal, o sistema de diagnóstico ao receber uma mensagem de falha desse vértice verifica quais os vértices que se encontram a essa distância. Este processo repete-se para cada vértice em falha, independentemente de estar ou não dentro do raio de outro vértice já em falha. Para guardar a informação dos vértices que estão dentro do mesmo raio, definiu-se a estrutura *NetworkFault* (Tabela 14).

NetworkFault	
State	- *Open*: indica que pode receber novas falhas - *Close*: indica que a rede de falha está pronta para ser diagnosticada
ConnectionType	Contém o tipo de interacção da rede de falha (Eléctrica, Fluxo, etc)
Network	Vector que contém todos os vértices dentro da distância
SourceNode	Contém o vértice de origem (*Node*)
Distance	Indica a distância da rede de falhas
Symptoms	Contém todos os vértices em falha dentro da distância

Tabela 14 – Estrutura *NetworkFault*

No exemplo da Figura 26, a estrutura *NetworkFault* correspondente seria equivalente à da Tabela 15.

NetworkFault	
State	*Open*
ConnectionType	Eléctrica
Network	*Conveyor3*, *Conveyor2*, *Conveyor5*, *Robot2*, *Conveyor1*, *Conveyor4*, *Conveyor6* e *Robot3*
SourceNode	*Node* correspondente ao *Conveyor3*
Distance	2
Symptoms	*Conveyor3*

Tabela 15 – Estrutura *NetworkFault* do exemplo da **Figura 26**

Para cada mensagem de falha é criada uma estrutura *NetworkFault*, em que o único campo que permanece vazio é o de *Symptoms*. Este campo corresponde aos vértices que estão em falha (enviaram uma mensagem de falha) e que se encontram na *Network*.

O objectivo deste processo, é reunir toda a informação em torno do vértice central (*SourceNode*) e usar, posteriormente, essa informação para fazer o diagnóstico. Para tal, utilizou-se o *Drools* recorrendo aos factos e às regras (descritos de seguida). Para além da estrutura *NetworkFault* é, também, criada uma estrutura *Symptom* (Tabela 16) que contém a informação da mensagem de falha recebida.

Symptom	
Name	Nome do vértice em falha
ConnectionType	Tipo de falha (Eléctrica, Fluxo, etc)

Tabela 16 – Estrutura *Symptom*

As duas estruturas (*NetworkFault* e *Symptom*) são introduzidas como factos na *Working Memory* e interagem com as seguintes regras inseridas na *Production Memory*:

```
rule "Verify new symptom for the existing networkfaults"
   when
        s : Symptom(n : nodeName, t1 : type)
        nf : NetworkFault(status == NetworkFault.Open, t2 : type, sn: sourceNode, network : network)
        eval(t2.equals(t1))
        eval(!sn.getName().equals(n))
        eval(network.contains(n))
    then
        nf.addSymptom(s);
end
```

Figura 27 – Regra para introduzir uma falha na *NetworkFault*

A regra presente na Figura 27 é utilizada para relacionar uma nova falha (*Symptom*) com as redes de falhas (*NetworkFault*) existentes. Quando um novo *Symptom* é introduzido, a regra é testada para todas as estruturas *NetworkFault* existentes, verificando se o nome do vértice em falha (campo *Name* da estrutura *Symptom*) está incluído na *Network* de cada *NetworkFault* sendo adicionado ao campo *Symptoms*. A estrutura *Symptom* está definida como evento no *Drools*, o que faz com que depois de todas regras serem testadas, a estrutura seja automaticamente removida.

Na arquitectura propôs-se que o sistema esperasse um determinado tempo sempre que é introduzida uma nova falha na *NetworkFault* (*Symptoms*), ou seja sempre que existe uma possível propagação de falha. Mas, como o *Drools* (nas versões existentes até ao momento) apresenta limitações tal não é possível. Assim, o *State* de uma *NetworkFault* é alterado para *Close* ao fim de um determinado tempo presente na *Working Memory*, sendo introduzida numa *HashMap* e removida do *Drools*.

Para obrigar o *Drools* a esperar um tempo pré-definido, utilizou-se uma condição que nunca será verdadeira, ou seja, a existência de uma estrutura *Node* inserida na *Working Memory*. A regra responsável pelo processo descrito anteriormente está representada na Figura 28.

```
rule "Close networkfault"
    when
        nf : NetworkFault()
        not( Node(this after[0s, 5s] nf))
    then
        ((Vector)droolsNetworkFaults.get(nf.getConnectionType())).add(nf);
        retract(nf);
end
```

Figura 28 – Regra para alterar o *State* de *Open* para *Close* de uma *NetworkFault*

A *HashMap droolsNetworkFaults* contém um vector de *NetworkFaults* para cada tipo de interacção presente na rede de vértices e cada *NetworkFault* é adicionada ao vector que representa o tipo de falha correspondente. Considerando o exemplo da Figura 26 e assumindo que após a mensagem de falha recebida do *Conveyor3* também o *Conveyor5* e o *Conveyor6* enviaram mensagens de falhas (Figura 29), a *HashMap droolsNetworkFaults* contém no respectivo vector três estruturas *NetworkFault* representadas nas Tabelas 17, 18 e 19.

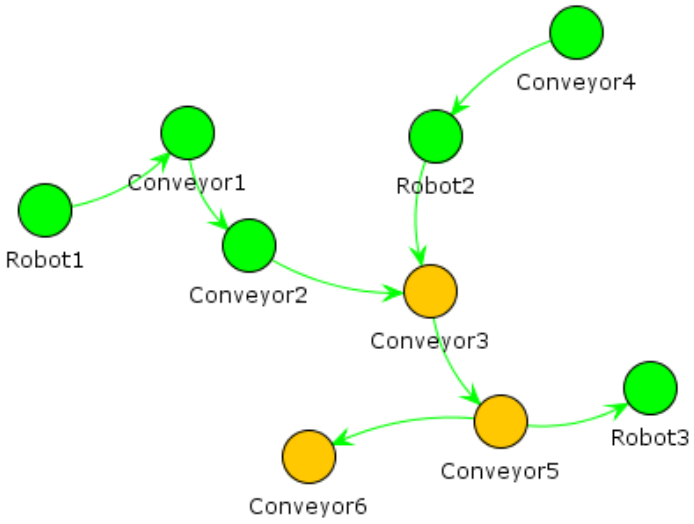

Figura 29 – Exemplo de falha no *Conveyor3*, *Conveyor5* e *Conveyor6*

NetworkFault	
State	*Close*
ConnectionType	Eléctrica
Network	*Conveyor3*, *Conveyor2*, *Conveyor5*, *Robot2*, *Conveyor1*, *Conveyor4*, *Conveyor6* e *Robot3*
SourceNode	*Node* correspondente ao *Conveyor3*
Distance	2
Symptoms	*Conveyor3*, Conveyor5 e *Conveyor6*

Tabela 17 – Estrutura *NetworkFault* do *Conveyor3*

NetworkFault	
State	*Close*
ConnectionType	Eléctrica
Network	*Conveyor5*, *Conveyor6*, *Conveyor3*, *Robot3*, *Conveyor2* e *Robot2*
SourceNode	*Node* correspondente ao *Conveyor5*
Distance	2
Symptoms	*Conveyor5* e Conveyor6

Tabela 18 – Estrutura *NetworkFault* do *Conveyor5*

NetworkFault	
State	*Close*
ConnectionType	Eléctrica

Network	*Conveyor6*, *Conveyor3*, *Conveyor5* e *Robot3*
SourceNode	*Node* correspondente ao *Conveyor6*
Distance	2
Symptoms	*Conveyor6*

Tabela 19 – Estrutura *NetworkFault* do *Conveyor6*

Quando uma estrutura *NetworkFault* é inserida no vector (e é removida da *Working Memory*) o sistema verifica se as falhas (*Symptoms*) existentes nessa estrutura ainda são válidas, ou seja se ainda estão presentes na rede. Para tal, o sistema verifica se o *SensorStatus* de cada um dos vértices em *Symptoms* ainda assinala a falha. Esta filtragem de informação é necessária visto que as falhas podem já ter sido diagnosticadas e, como tal, devem ser descartadas e removidas da *NetworkFault* antes de ser realizado o diagnóstico.

4.1.4. DIAGNÓSTICO

O sistema verifica, periodicamente, se existem falhas para serem analisadas e diagnosticadas, sendo a explicação da falha do vértice de origem e possíveis propagações desta o objectivo do diagnóstico. Tendo isto em conta, o primeiro passo é identificar, entre todos os vértices presentes na *Network* da *NetworkFault*, quais os que podem (em caso de falha) afectar o vértice de origem. Recorrendo às funcionalidades do *JUNG*, que disponibiliza um mecanismo de pesquisa de distâncias entre vértices, verifica-se para todos os vértices da rede que conseguem atingir o vértice de origem.

A Figura 30 exemplifica uma rede de falha, encontrando-se na Tabela 20 a respectiva *NetworkFault*. Sendo o *Conveyor3* o *SourceNode* da *NetworkFault*, os únicos vértices da *Network* que conseguem alcançar o *Conveyor3* são o *Conveyor1* e o *Conveyor2*.

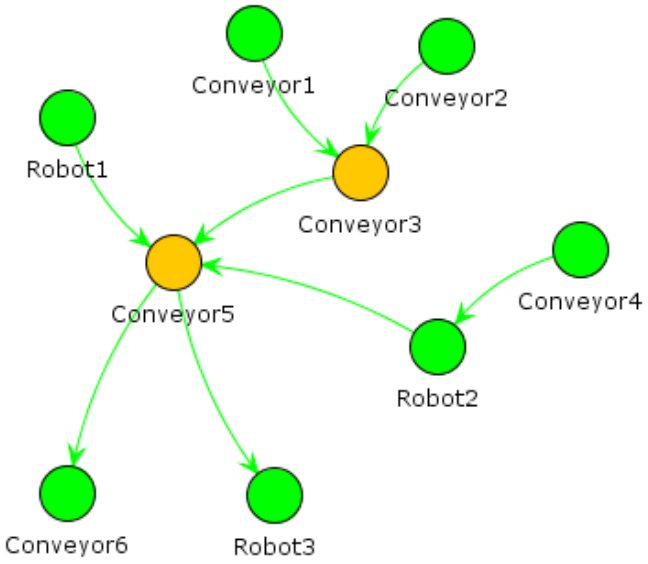

Figura 30 – Exemplo de uma rede de falha com o *Conveyor3* como vértice de origem

NetworkFault	
State	*Close*
ConnectionType	Eléctrica
Network	*Conveyor3*, *Conveyor2*, *Conveyor1*, *Conveyor5*, *Conveyor6*, *Robot1*, *Robot3* e *Robot2*
SourceNode	*Node* correspondente ao *Conveyor3*
Distance	2
Symptoms	*Conveyor3* e *Conveyor5*

Tabela 20 – Estrutura *NetworkFault* do exemplo da Figura 30

Depois de identificados os vértices que podem alcançar a origem (incluindo a própria origem), o sistema vai analisar todas as possibilidades de propagação de falhas para tentar explicar a causa da falha do *Conveyor3*. Essa falha pode ser o resultado de uma propagação do *Conveyor1* ou do *Conveyor2* (ou até de ambos) ou pode, ainda, ser uma falha iniciada pelo próprio *Conveyor3*. Em qualquer uma das situações anteriores, os restantes vértices da rede e as suas falhas são também alvo de análise e diagnóstico.

O sistema de diagnóstico tenta, a partir de cada um dos vértices que podem alcançar a origem (denominados como vértices iniciais), construir uma rede de propagação de falha que inclua o vértice de origem. Para tal e recorrendo às funcionalidades do *JUNG*, o sistema reproduz através da rede de vértices da *NetworkFault* todas as sub-redes possíveis tendo como origem cada um dos vértices iniciais. Os vértices

de cada uma das sub-redes são analisados e diagnosticados sendo atribuída a respectiva classificação proposta na arquitectura.

O processo de análise e diagnóstico proposto recorre a factos históricos para diagnosticar as falhas, factos esses que representam a possível existência ou inexistência da propagação de um tipo de falha (Eléctrica, Fluxo, etc.) entre dois tipos de vértices (*Conveyor*, *Robot*, *Gripper*, etc). A Tabela 21 representa a estrutura *Fact* que é utilizada para definir um facto histórico sobre uma determinada falha.

Fact	
State	- *Waiting*: indica que o facto ainda é temporário e não pode ser utilizado para diagnóstico - *Confirm*: indica que o facto pode ser utilizado para diagnóstico
SourceType	Tipo do vértice da origem
DestType	Tipo do vértice do destino
Transmit	Indica se existe propagação de uma falha
ConnectionType	Indica o tipo de falha que o facto representa
Confirm	Contém o número de vezes em que este facto foi confirmado
NotConfirm	Contém o número de vezes que o facto não foi confirmado
Threshold	Representa a percentagem após a qual o facto passa a ser válido no caso do estado *Waiting* e a percentagem após a qual o facto deixa de ser válido no caso do estado *Confirm* sendo o campo Transmit alterado

Tabela 21 – Estrutura *Fact*

Para fazer a gestão dos factos históricos, utilizou-se o *Drools* sendo as estruturas *Fact* inseridas na *Working Memory* em representação do conhecimento sobre a propagação de falhas entre os elementos (vértices) da rede. Para poder interagir com este conhecimento criou-se a estrutura *TestFact* (Tabela 22). Ao introduzir esta estrutura na *Working Memory* é possível consultar, inserir e modificar a informação presente no sistema sobre a propagação de falhas entre dois tipos de vértices da rede.

TestFact	
State	- *Test_Confirm*: indica que pretende-se adquirir informação - *Confirm*: indica que pretende-se inserir informação nas estruturas *Fact* que tem o estado *Confirm* - *Waiting*: indica que pretende-se inserir informação nas estruturas *Fact* que tem o estado *Waiting* - Done: indica que o *TestFact* já foi testado
SourceType	Tipo do vértice da origem

DestType	Tipo do vértice do destino
Transmit	Utilizado para os estados *Confirm* e *Waiting* e indica a condição a testar (propagar ou não propagar)
ConnectionType	Indica o tipo de falha a ser testada
Result	Informação sobre o resultado do teste - Para o estado *Test_Confirm*: -1: indica que não existe informação sobre a propagação de falha entre o tipo de origem e o tipo de destino 0: indica que não existe propagação de falha entre os dois tipos de vértice 1: indica que existe propagação de falha entre os dois tipos de vértice - Para os estados *Confirm* e *Waiting*: -1: indica que não existe informação sobre a propagação de falha entre o tipo de origem e o tipo de destino 0: indica que o teste à propagação (*Transmit*) entre os dois tipos de vértice não se verifica 1: indica que o teste à propagação entre os dois tipos de vértice verifica-se

Tabela 22 – Estrutura TestFact

Para haver uma interacção entre as estruturas *Fact* e as estruturas *TestFact* procedeu-se à elaboração da regra presente na Figura 31.

```
rule "Test Test_Confirm"
    when
        tf : TestFact(status == TestFact.Test_Confirm, st : sourceType,
            dt : destType, cttf : connectionType)
        f : Fact(status == Fact.Confirm, connectionType == cttf, sourceType == st,
            destType == dt, t : transmit)
    then
        tf.setStatus(TestFact.Done);
        if(t){
            tf.setResult(1);
        }else{
            tf.setResult(0);
        }
        update(tf);
end
```

Figura 31 – Regra para identificar uma propagação de falha

A regra da Figura 31 é utilizada para adquirir a informação sobre a propagação de um determinado tipo de falha entre dois tipos de vértices. Ao inserir um *TestFact* todas as estruturas *Fact* existentes na *WorkingMemory* são testadas tendo em conta os critérios

definidos pela regra. Em caso de sucesso, a existência (ou inexistência) da propagação é inserida no *Result* e o *State* do *TestFact* é alterado para *Done*. A partir do *Result* e do *State*, o sistema consegue determinar se os tipos de vértices em causa podem propagar falhas entre si, não propagam falhas entre si ou não é possível concluir nada sobre a propagação entre os dois tipos de vértices.

Para analisar cada sub-rede, o sistema de diagnóstico aplica a regra da Figura 31 ao vértice inicial e aos seus vizinhos. No caso de se confirmar a propagação para um vizinho, a mesma regra é aplicada para esse vértice e os seus vizinhos. Quando o *Result* do *TestFact* indicar que um vértice pode estar a propagar uma falha a outro vértice mas o sensor deste não indica nenhuma falha, o mesmo processo de análise é feito para os vizinhos desse vértice. Deste modo, é possível o sistema identificar falhas em vértices mesmo quando os seus sensores não assinalam falha.

A Figura 32 demonstra um exemplo em que o *Robot1* e o *Conveyor2* assinalam falha e o *Conveyor1* apesar de não assinalar falha, através do seu sensor, encontra-se também em falha. Assumindo as estruturas *Fact* das Tabelas 23 e 24, o sistema ao analisar o *Robot1* conclui, através do *Result* do *TestFact*, que este pode propagar falhas ao *Conveyor1*. Para confirmar esta situação, o sistema analisa os vizinhos do *Conveyor1* verificando que o *Conveyor2* se encontra em falha e que o *Conveyor1*, nesta situação, pode propagar-lhe a falha. O sistema de diagnóstico encara esta situação como uma falha no sensor do *Conveyor1* e assume falha neste.

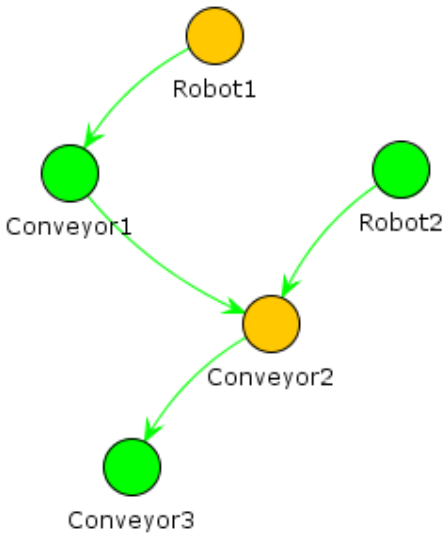

Figura 32 – Exemplo de falha do sensor no *Conveyor1*

Fact1	
Estado	*Confirm*
Tipo de Origem	*Robot*
Tipo de Destino	*Conveyor*
Propagação	True
Tipo de falha	Electrica

Tabela 23 - Estrutura *Fact1* do exemplo da Figura 32

Fact2	
Estado	*Confirm*
Tipo de Origem	*Conveyor*
Tipo de Destino	*Conveyor*
Propagação	True
Tipo de falha	Electrica

Tabela 24 – Estrutura *Fact2* do exemplo da Figura 32

A Figura 32 ilustra um exemplo de uma propagação de falha que pode verificar-se na realidade, é possível haver dependências eléctricas entre os componentes. Face a isto, uma falha eléctrica num dos componentes pode, eventualmente, provocar uma falha eléctrica num componente que seja seu vizinho.

Depois de todos os vértices da sub-rede serem analisados, o módulo de diagnóstico verifica se o diagnóstico obtido consegue explicar a falha do vértice de origem.

No caso de haver mais que uma sub-rede a conseguir explicar a falha do vértice de origem, a selecção é feita tendo em conta a classificação de cada uma. A classificação é construída à medida que a respectiva sub-rede é analisada e obedece à expressão apresentada na arquitectura do sistema.

O diagnóstico escolhido é o que tiver a maior classificação, desde que esta seja superior a um valor mínimo (configurável). Caso seja inferior, é escolhido o diagnóstico que contém, apenas, a falha do vértice de origem.

Para guardar toda a informação sobre o diagnóstico de uma sub-rede foi definida a estrutura *DiagnosticNetworkFault*, representada na Tabela 25.

DiagnosticNetworkFault	
Rank	Indica a classificação atribuída pelo sistema de diagnóstico
ConnectionType	Contém o tipo de interacção (Eléctrica, Fluxo, etc)
Network	Vector que contém todos os vértices da sub-rede

SourceNode	Contém o vértice que está na origem da falha
SymptomNodes	Vector que contém todos os vértices em falha
Graph	Variável do tipo *graph* que contém a representação dos vértices em falha (vértices e arcos)

Tabela 25 – Estrutura DiagnosticNetworkFault

A variável *Grafo* contém a representação gráfica da rede de falha que apresenta os vértices e os arcos em falha. Partindo do vértice que está a provocar a falha (*SourceNode*) é possível representar visualmente a falha e as suas propagações.

Na Figura 33 encontra-se um exemplo (adaptado da Figura 32) em que é obtido um diagnóstico que indica que o *Robot1* está na origem de uma falha, que se propaga ao *Conveyor1* e consequentemente ao *Robot2*. Como foi dito atrás, o sensor do Conveyor1 (assinalado com uma circunferência magenta) não detecta nenhuma falha, no entanto, através da informação sobre as falhas passadas, o diagnóstico obtido indica que se encontra em falha.

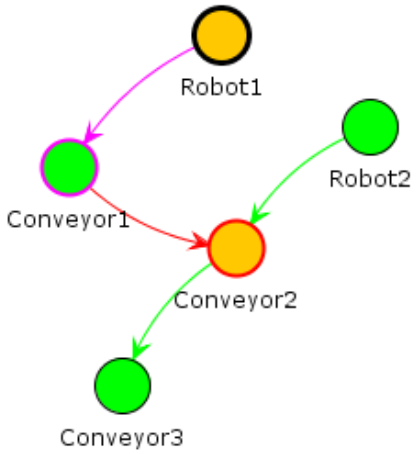

Figura 33 – Exemplo de diagnóstico na ausência de falha de sensor

A estrutura *DiagnosticNetworkFault* correspondente, que contém toda a informação sobre o diagnóstico, encontra-se na Tabela 26.

DiagnosticNetworkFault	
Rank	66,6%
ConnectionType	Eléctrica
Network	*Robot1*, *Robot2*, *Conveyor1*, *Conveyor2*, *Conveyor3*
SourceNode	*Robot1*
SymptomNodes	*Robot1*, *Conveyor1*, *Conveyor2*
Graph	…

Tabela 26 – Estrutura que contém o diagnóstico representado na Figura 33

4.1.5. APRENDIZAGEM

Através da variável *Graph*, da respectiva estrutura *DiagnosticNetworkFault* do diagnóstico seleccionado, obtém-se as interacções por onde a falha está se está ou não a propagar. Para cada uma dessas interacções é introduzida na *Working Memory* uma estrutura *TestFact* com a informação relevante à propagação entre os dois tipos de vértices da interacção. O campo *State* é preenchido com o valor *Confirm* e o *Transmit* com o valor *True* ou *False*, consoante o estado da interacção. A regra para interagir com estas definições da estrutura *TestFact* encontra-se na Figura 34.

```
rule "Confirm fact"
salience 98
    when
        tf : TestFact(status == TestFact.Confirm, st : sourceType, dt : destType,
            t1 : transmit, cttf : connectionType)
        f : Fact(status == Fact.Confirm, connectionType == cttf, sourceType == st,
            destType == dt, t2 : transmit)
then
        tf.setStatus(TestFact.Done);
        if(t1 == t2){
            tf.setResult(1);
            f.incConfirm();
        }else{
            tf.setResult(0);
            f.incNotConfirm();
        }
        update(tf);
        update(f);
end
```

Figura 34 – Regra para testar uma estrutura *Fact* com o *State Confirm*

Se existir na *Working Memory* do Drools uma estrutura *Fact* com o State igual a *Confirm* entre os dois tipos de vértices e que confirme o campo *Transmit*, então é incrementado o valor do campo *Confirm*. Caso contrário, é incrementado o valor do campo *NotConfirm*. Se não existir uma estrutura *Fact*, que defina a informação sobre a propagação entre os dois tipos de vértice, o valor do *Result* será -1 (o valor por omissão da estrutura). Ao acontecer isto, é introduzida uma estrutura *TestFact* igualmente preenchida mas com o *State* em *Waiting*, que ao interagir com a regra da Figura 35 permite aferir se existe alguma estrutura *Fact* que verifique as condições.

```
rule "Waiting fact"
salience 98
    when
        tf : TestFact(status == TestFact.Waiting, st : sourceType, dt : destType,
            t1 : transmit, cttf : connectionType)
        f : Fact(status == Fact.Waiting, connectionType == cttf, sourceType == st,
            destType == dt, t2 : transmit)
then
        tf.setStatus(TestFact.Done);
        if(t1 == t2){
            tf.setResult(1);
            f.incConfirm();
        }else{
            tf.setResult(0);
            f.incNotConfirm();
        }
        update(tf);
        update(f);
end
```

Figura 35 – Regra para testar uma estrutura *Fact* com o *Estado Waiting*

Se o *Result* do último *TestFact* continuar com o valor -1, significa que não existe nenhuma informação temporária sobre a propagação entre os dois tipos de vértices. Perante esta situação, o sistema de diagnóstico introduz na *Working Memory* uma estrutura *Fact* com a informação do estado da interacção e com o State igual a *Waiting*.

Quando todas as interacções estão analisadas, as regras das Figuras 36 e 37 são accionadas ao serem introduzidas duas *String* com os valores *TestConfirm* e *TestWaiting*. No caso das estruturas *Fact* com o *State* igual a *Confim*, verifica-se se a percentagem do *NotConfirm* ultrapassou o campo *Threshold*. No caso de se confirmar, o valor do *Transmit* é alterado. Nas estruturas *Fact* com o *State* igual a *Waiting*, são feitos os mesmos cálculos usando o valor absoluto e sendo o State alterado de *Waiting* para *Confirm*.

```
rule "Test confirm"
salience 96
    when
        s : String()
        eval(s.equals("Test confirm"))
        f : Fact(status == Fact.Confirm, th : threshold, nc : notConfirm, c : confirm)
        eval((double)nc > ((double)(th*c)/100))
    then
        f.setTransmit(!f.getTransmit());
        f.setNotConfirm(0);
        f.setConfirm(1);
        update(f);
end
```

Figura 36 – Regra para avaliar uma estrutura *Fact* com o *Estado Confirm*

```
rule "Test waiting"
salience 96
    when
        s : String()
        eval(s.equals("Test waiting"))
        f : Fact(status == Fact.Waiting, th : threshold, nc : notConfirm, c : confirm)
    then
        if(c >= th){
            f.setNotConfirm(0);
            f.setConfirm(1);
            f.setStatus(Fact.Confirm);
            f.setThreshold(20);
            update(f);
        }else if(nc > th){
            f.setTransmit(!f.getTransmit());
            f.setNotConfirm(0);
            f.setConfirm(1);
            f.setStatus(Fact.Confirm);
            f.setThreshold(20);
            update(f);
        }
end
```

Figura 37 – Regra para avaliar uma estrutura *Fact* com o *Estado Waiting*

Quando a opção 3 da Figura 20 está activada, o sistema de diagnóstico recebe durante um determinado número de falhas, toda a informação sobre estas e a respectiva propagação. No modo *Training* o mecanismo de aprendizagem baseia-se na utilização das regras anteriormente explicadas e nas estruturas *Fact*, que são definidas e introduzidas na *Working Memory* ao receber a informação da propagação de cada falha.

Durante esse intervalo de tempo o sistema não realiza nenhum diagnóstico e limita-se a aprender com a informação da propagação das falhas. Quando o intervalo de tempo termina são accionadas as regras para avaliar as estruturas *Fact* inseridas.

O fluxograma que se segue (Figura 38) caracteriza o sistema de aprendizagem que foi aplicado no módulo *Aprendizagem* sempre que um diagnóstico é finalizado.

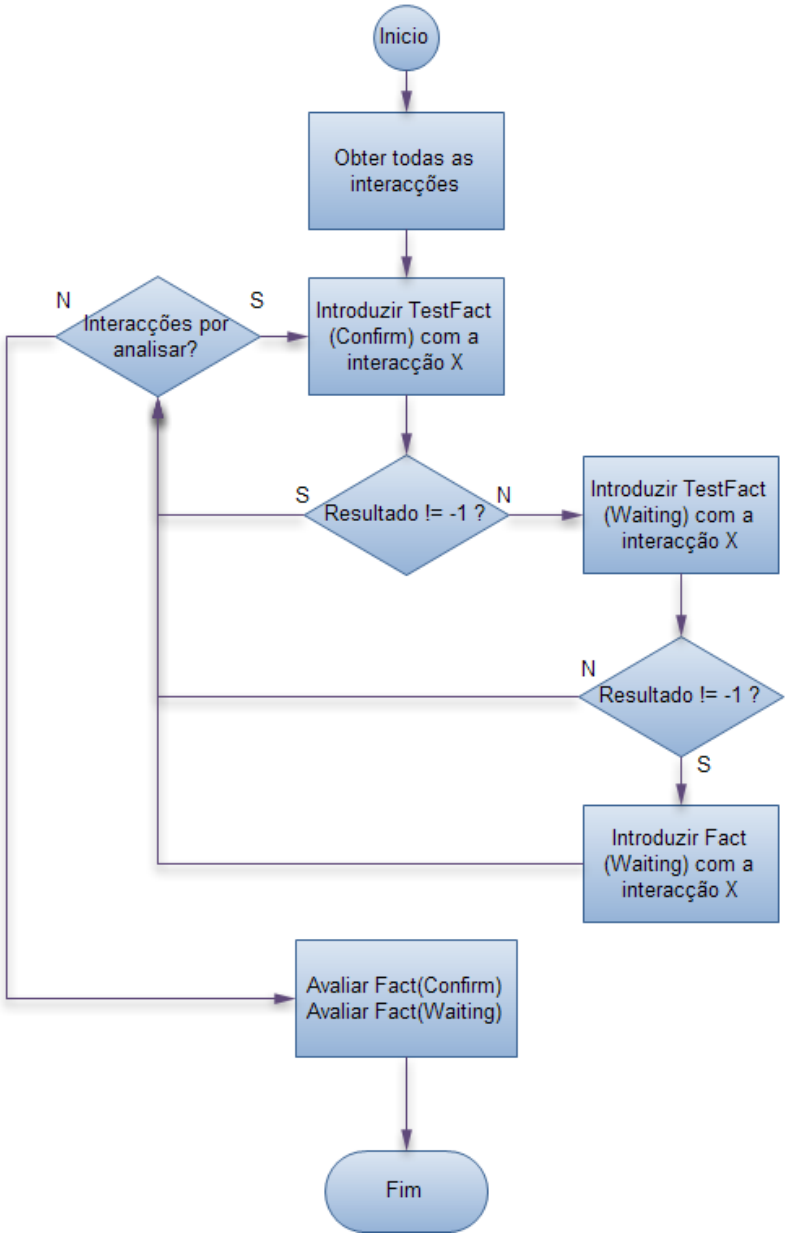

Figura 38 – Fluxograma de funcionamento do sistema de aprendizagem

5. TESTES E RESULTADOS

Neste capítulo é apresentada a plataforma de testes utilizada para comparar os sistemas de diagnósticos e que utiliza a complexidade da rede formada pelos componentes e interacções entre estes. São ainda apresentados os testes realizados e os resultados obtidos pelos dois sistemas de diagnóstico em estudo.

5.1. PLATAFORMA DE TESTES

A plataforma de testes permite a criação de redes de componentes e interacções, com uma determinada complexidade, que são transmitidas aos sistemas de diagnóstico em estudo. Posteriormente, a propagação de falhas é activada, originando falhas nos componentes que os sistemas de diagnóstico processam. Depois de diagnosticar as falhas, os resultados de ambos os sistemas de diagnóstico são registados na plataforma de testes para poderem ser analisados. Esta análise é feita recorrendo ao estado verdadeiro do sistema (estado real da propagação de falhas no sistema), que é também registado na altura da recepção dos resultados de ambos os sistemas de diagnóstico.

A criação da rede é feita tendo em conta a quantidade e o tipo de componentes pretendidos (*Conveyor*, *Robot*, etc.) e a conectividade média pretendida entre estes. Os componentes são criados e enviados aos sistemas de diagnóstico e, posteriormente, são criadas as interacções entre estes, sendo também enviadas aos sistemas de diagnóstico. Para criar as interacções, acede-se á matriz de adjacência e em cada uma das linhas preenche-se, aleatoriamente, um número de entradas igual à conectividade média pretendida (complexidade), como indicado em [72].

Para criar uma falha na rede é necessário injectar uma falha num dos componentes, escolhido aleatoriamente, sendo essa falha propagada através do modelo de propagação.

O modelo de propagação utilizado é probabilístico, ou seja, quando um componente recebe uma falha de outro componente, através das suas interacções de

entrada, tem uma determinada probabilidade de ficar afectado. Esta probabilidade pode ter dois valores e depende do tipo de vulnerabilidade atribuída ao componente. Durante criação da rede, é definida uma percentagem de componentes da rede que vão ser dados como vulneráveis, sendo a restante parte dada como não vulnerável. A esta percentagem dá-se o nome de vulnerabilidade da rede.

A cada um destes conjuntos de componentes, é possível atribuir um valor que representa a probabilidade que cada um tem de ser afectado por uma falha. Por defeito, este valor é 95% para os componentes vulneráveis e 20% para os não vulneráveis. A escolha destes valores é feita tendo em conta que se pretende que um componente, dado como vulnerável, tenha um grande impacto na propagação das falhas.

Quando um componente, vulnerável ou não vulnerável, está em falha, seja porque aceitou uma falha ou porque o próprio provocou a falha, esta é sempre propagada aos seus componentes vizinhos através das suas interacções de saída.

No sentido de testar o comportamento de ambos os sistemas de diagnóstico, na presença de falhas do próprio "sensor" de cada componente, introduziu-se uma probabilidade em cada componente que representa a hipótese deste não detectar que está em falha.

Assim, quando um componente fica afectado por uma falha, e o seu sensor não a detecta, esta será transparente para os sistemas de diagnóstico. O valor da probabilidade de falha no sensor é igual para todos os componentes e é configurável.

Deste modo torna-se possível criar condições de teste em que, no pior cenário, não existe nenhuma informação "sensorial" associada aos componentes.

5.1.1. VALIDAÇÃO

Para validar a plataforma de testes e caracterizar o comportamento do modelo de propagação de falhas, procedeu-se à realização de uma série de testes em diferentes condições.

Foram criadas redes de 25, 50 e 75 componentes, que foram submetidas a baterias de propagação de falhas, combinando diferentes medidas de complexidade e diferentes percentagens de vulnerabilidade. As probabilidades de um componente aceitar

a falha, consoante seja vulnerável ou não, são 95% e 20%, respectivamente, e mantiveram-se constantes durante os testes.

Os seguintes gráficos representam a percentagem de componente da rede afectados, com a variação da vulnerabilidade atribuída à rede.

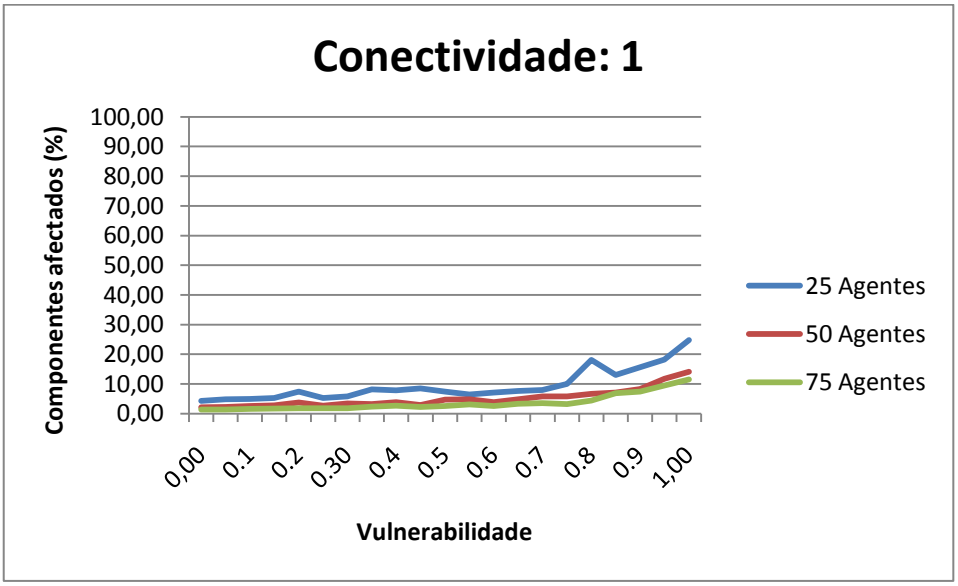

Figura 39 – Propagação de falhas numa rede de complexidade 1

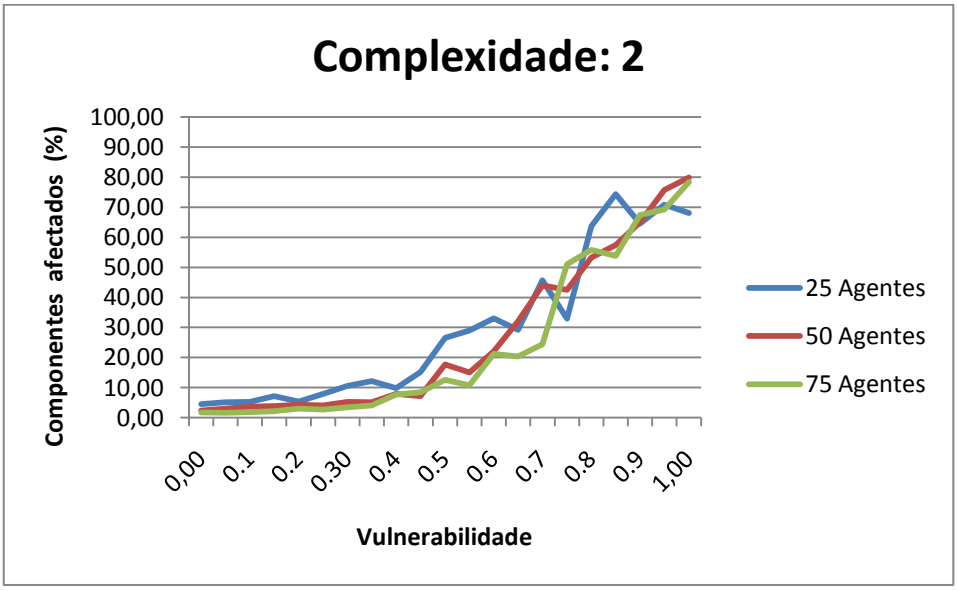

Figura 40 – Propagação de falhas numa rede de complexidade 2

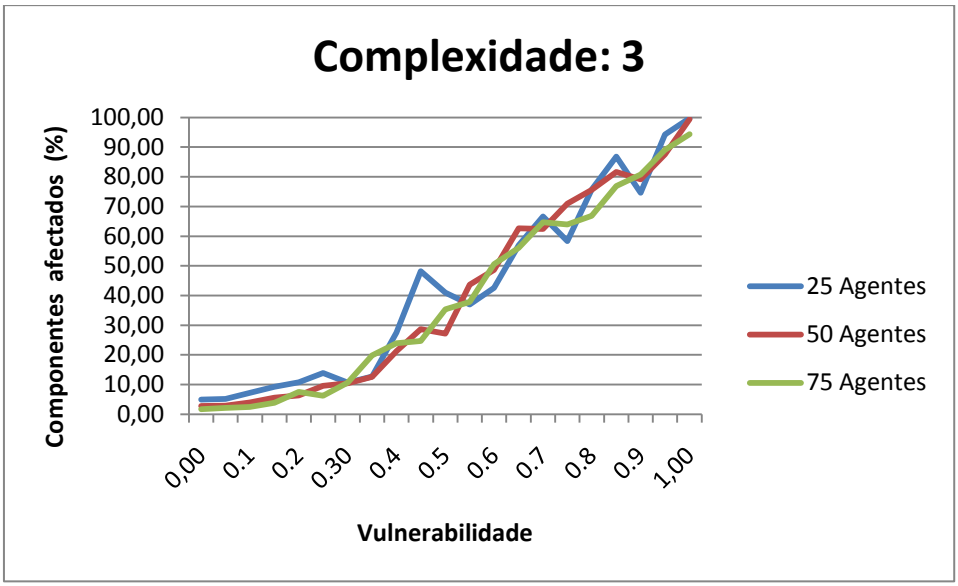

Figura 41 – Propagação de falhas numa rede de complexidade 3

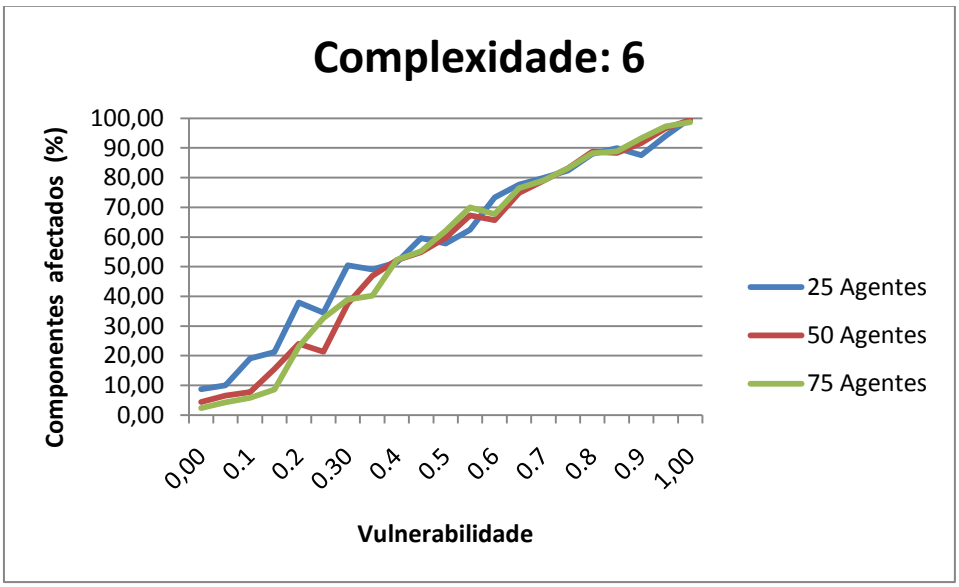

Figura 42 – Propagação de falhas numa rede de complexidade 6

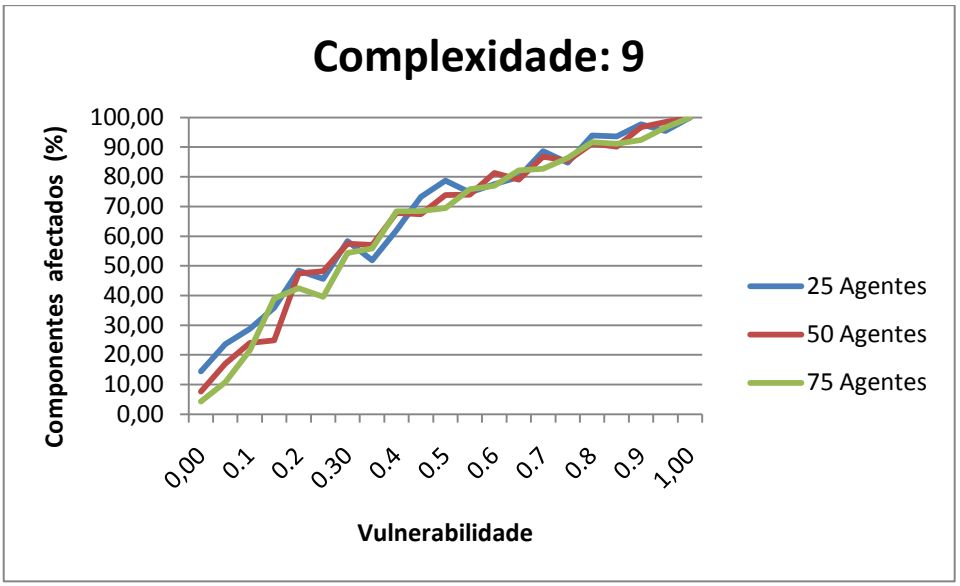

Figura 43 – Propagação de falhas numa rede de complexidade 9

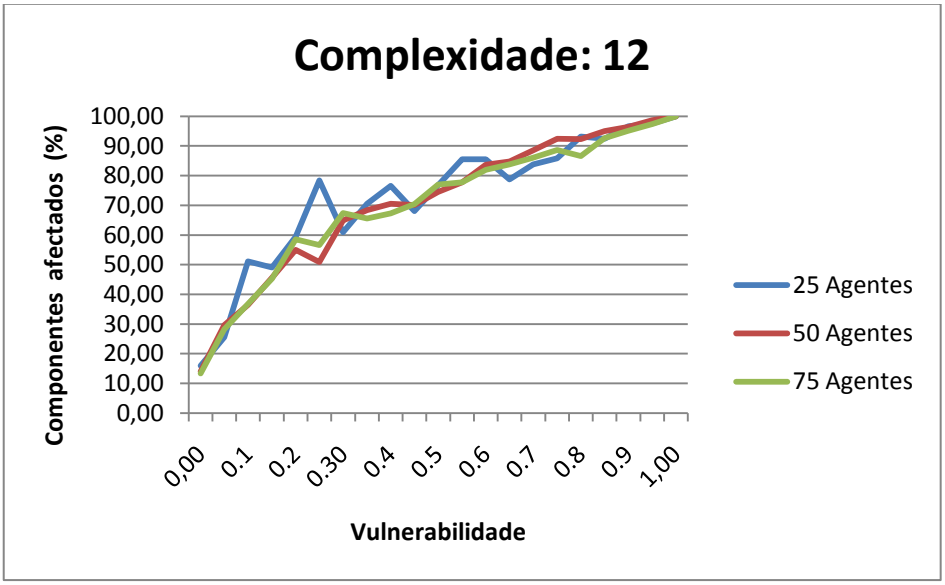

Figura 44 – Propagação de falhas numa rede de complexidade 12

A partir dos gráficos anteriores conclui-se que o efeito do modelo de propagação de falhas é independente do número de componentes da rede, e depende apenas dos valores da complexidade e da vulnerabilidade utilizados. A partir de um valor de complexidade 3, o comportamento da propagação passa de uma aproximação linear a um logaritmo, a partir do qual o número de componentes em falha aumenta bastante mesmo para uma complexidade baixa. Deste modo, torna-se possível realizar testes adequados

em redes com diferentes dimensões, assegurando-se uma propagação de falhas controlada.

É de salientar que com esta plataforma de testes, é possível avaliar correctamente o desempenho de qualquer sistema de diagnóstico, desde que o sistema físico ou lógico, a diagnosticar, possa ser representado e absorvido pela plataforma numa perspectiva de rede.

5.2. RESULTADOS

Os dois sistemas de diagnóstico foram testados em dois cenários diferentes, na célula NOVAFLEX (situada na UNINOVA, Portugal) e em redes aleatórias de componentes geradas pela plataforma de testes. Em ambos os casos, os resultados de cada sistema de diagnóstico, para cada simulação de falha, são comparados com a realidade obtendo-se uma percentagem que representa a performance do diagnóstico.

Através de várias simulações de falhas, é possível calcular para um determinado intervalo de confiança, a performance do sistema de diagnóstico (média das performances em cada simulação) e os respectivos desvios associados ao intervalo de confiança.

5.2.1. CÉLULA NOVAFLEX

A célula NOVAFLEX (Figura 45) é constituída por uma rede de *conveyors*, dois manipuladores industriais (robots) e várias *pallets* que circulam numa determinada ordem de modo a cumprir os seus planos. Cada um destes componentes é representado como um módulo de um EPS, que estabelece interacções com outros componentes, cooperando e organizando-se de forma a cumprir os seus objectivos.

Em [71], foi demonstrado que mesmo para um simples processo, a abstracção de cada componente como um agente autónomo, interactivo e independente, resulta num considerável conjunto complexo de interacções.

Toda a infra-estrutura da célula NOVAFLEX está implementada através de agentes JADE.

Figura 45 – Célula NOVAFLEX

A natureza do diagnóstico, mais com foco nas interacções entre os componentes do que nas especificidades do próprio dispositivo, garante que os diagnósticos acompanhem as mudanças estruturais e lógicas na perspectiva do controlo do sistema.

Deste modo surgem várias redes de acordo com as seguintes características do sistema:

$$interacções \in \{mecânica, fluxo, eléctrica, comunicação, pneumática, hidráulica\}$$

$$dispositivos \in \{conveyor, fixture, gripper, manipulator\}$$

No caso de teste da NOVAFLEX, consideram-se dez *conveyors* mencionados como *ConvN*, um manipulador, uma *gripper* e uma *fixture* com os nomes *Scara*, *Gripper* e *Fixture* respectivamente.

Como exemplos, das redes de interacções entre estes componentes, encontram-se na Figura 46 para as interacções de fluxo e na Figura 47 para a mecânica.

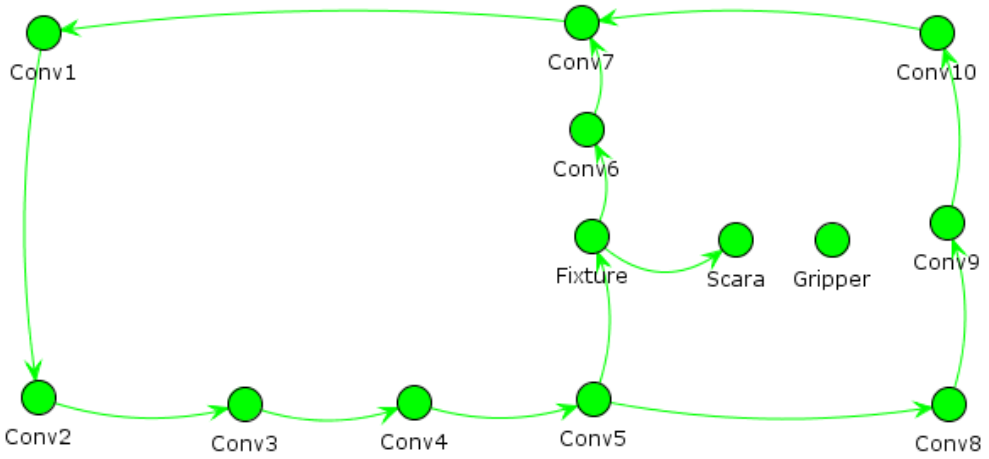

Figura 46 – Rede de fluxo da NOVAFLEX

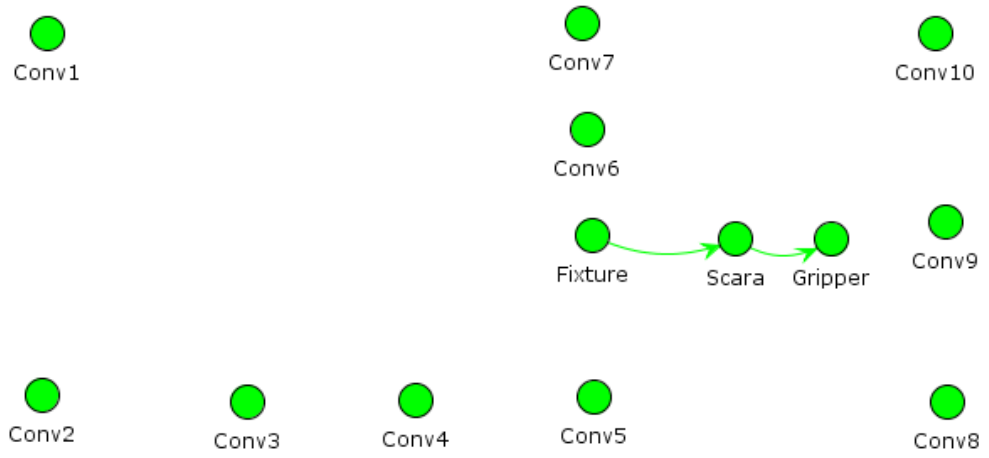

Figura 47 – Rede mecânica da NOVAFLEX

Os testes efectuados na NOVAFLEX centraram-se na rede de interacções de fluxo, já que é nesta que é estabelecida o maior número de interacções. Durante o funcionamento da célula foram simuladas várias falhas que representam as várias combinações de falhas possíveis entre os componentes da rede. Simulações como sequências de propagação de falhas entre componentes, falhas individuais dos componentes, falhas propagadas entre estes através de vários caminhos, etc.

Todas estas combinações resultaram em 60 simulações de cenários de falha num total de 300 falhas. A performance média apresentada para este conjunto de falhas e a variação da performance, tendo em conta um intervalo de confiança de 95%, encontra-se na seguinte tabela:

	Sistema centralizado	Sistema distribuído
Performance média (%)	92,63	93,17
Variação da performance (%)	±5,08	±4,80

Tabela 27 – Performance dos sistemas de diagnóstico na célula NOVAFLEX

Como se pode observar pela tabela anterior, ambos os sistemas de diagnóstico obtém resultados semelhantes e elevados nos testes efectuados na célula NOVAFLEX. A própria variação da performance é praticamente igual nas duas abordagens.

5.2.2. REDES GERADAS PELA PLATAFORMA DE TESTES

Foram realizados testes em redes de 50 componentes, este valor foi escolhido face ao número médio de componentes existentes nas células de manufactura industrial mas também face á capacidade computacional necessária para criar redes de maiores dimensões. A limitação computacional deve-se em parte, à necessidade de definir uma quantidade exorbitante de interacções numa rede de grandes dimensões, de modo a obter uma elevada complexidade.

Foram criadas duas redes através de plataforma de testes, que diferem na percentagem de componentes vulneráveis e que tem as seguintes características:

Rede de testes 1	
Número de componentes	50
Número de simulações de falhas	300
Componentes vulneráveis (%)	10
Probabilidade de afectação de componentes vulneráveis (%)	95
Probabilidade de afectação de componentes não vulneráveis (%)	20
Probabilidade de falha no sensor dos componentes (%)	10

Tabela 28 – Características da rede de testes 1

Rede de testes 2	
Número de componentes	50
Número de simulações de falhas	300
Componentes vulneráveis (%)	50
Probabilidade de afectação de componentes vulneráveis (%)	95
Probabilidade de afectação de componentes não vulneráveis (%)	20
Probabilidade de falha no sensor dos componentes (%)	10

Tabela 29 – Características da rede de testes 2

A escolha de diferentes percentagens de componentes vulneráveis, resulta da necessidade de avaliar o desempenho dos sistemas de diagnóstico perante cenários com uma maior e uma menor propagação de falhas. Para cada uma das redes de testes variou-se a complexidade com valores de 1, 2, 3, 6 e 9. Ou seja, a partir de cada uma das redes de teste, foram criadas outras cinco redes que foram usadas para testar os sistemas de diagnóstico.

Aplicando o modelo de propagação de falhas e os sistemas de diagnóstico a cada uma das redes, obtiveram-se os seguintes resultados:

	Sistema centralizado		Sistema distribuído	
Complexidade	Performance média (%)	Variação da performance (%)	Performance média (%)	Variação da performance (%)
1	90,3	±3,48	98,33	±1,37
2	61,68	±5,42	91,27	±2,79
3	61,99	±5,69	95,15	±2,18
6	34,49	±5,55	82,34	±2,58
9	16,72	±4,41	60,89	±2,95

Tabela 30 – Performance dos sistemas de diagnóstico na rede de testes 1

	Sistema centralizado		Sistema distribuído	
Complexidade	Performance média (%)	Variação da performance (%)	Performance média (%)	Variação da performance (%)
1	71,39	±5,07	94,41	±2,47
2	63,28	±3,76	84,30	±2,53
3	65,03	±2,51	78,45	±2,04
6	46,13	±1,06	68,56	±1,21
9	32,04	±3,22	62,46	±0,97

Tabela 31 – Performance dos sistemas de diagnóstico na rede de testes 2

As representações gráficas da performance dos sistemas de diagnóstico para as duas redes de teste, incluindo a variação da performance com um intervalo de confiança de 95%, encontram-se nas Figuras 48 e 49.

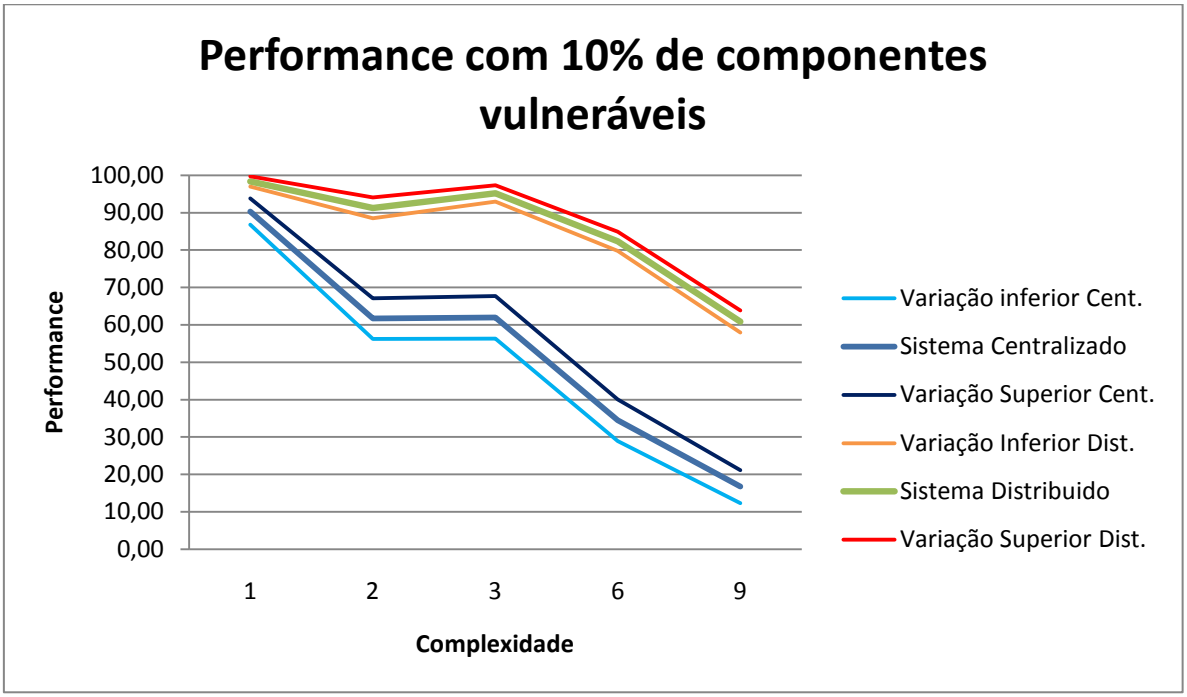

Figura 48 – Performance dos sistemas de diagnóstico na rede de testes 1

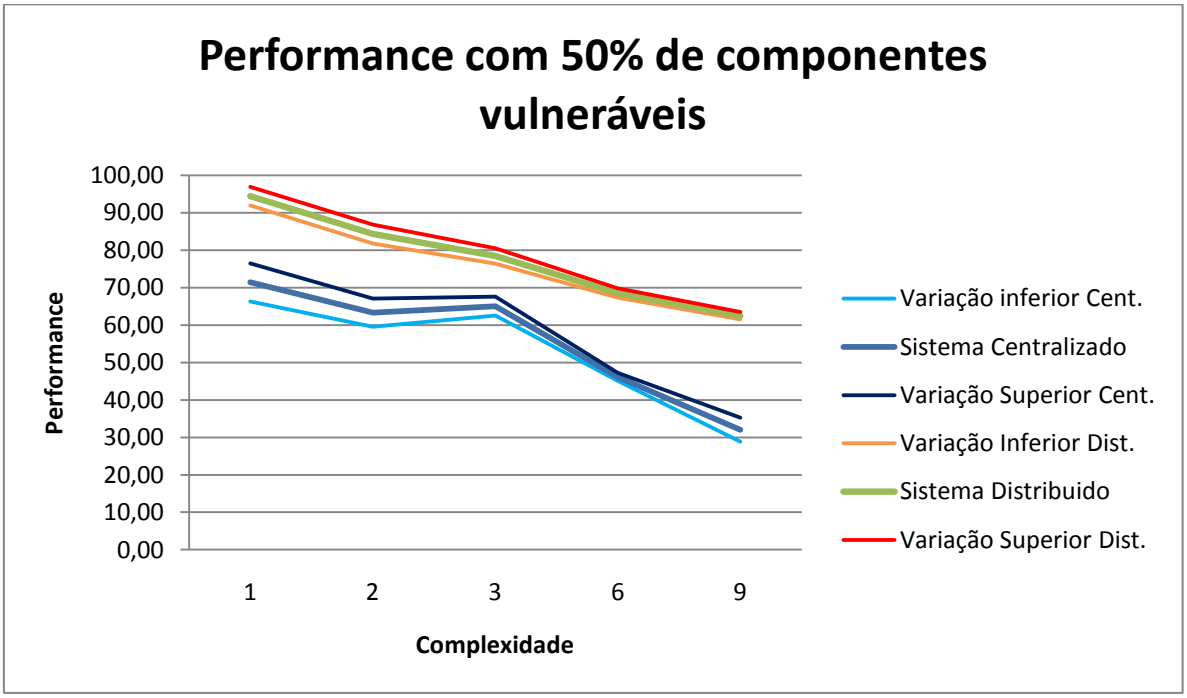

Figura 49 – Performance dos sistemas de diagnóstico na rede de testes 2

Quando os dois sistemas são sujeitos a condições de teste através da plataforma de testes, com a criação de redes com elevados graus de conectividade, os resultados obtidos são bastante diferentes.

O sistema de diagnóstico distribuído consegue uma performance muito superior ao sistema centralizado em ambas as redes de testes, nomeadamente na rede de testes 1, onde consegue uma performance constante até um grau de complexidade 3. O sistema centralizado consegue também uma performance constante entre a complexidade 2 e 3, mas mais baixa em relação á performance do sistema distribuído. A justificação para tal, deve-se em grande parte à quantidade e variedade de informação para analisar, o que gera ambiguidade no processo de relacionar a informação sobre as falhas actuais com a informação aprendida pelo sistema.

A performance de ambos os sistemas de diagnóstico tende a baixar, a partir do valor de complexidade 3, visto que é a partir dessa complexidade que o comportamento do modelo de propagação de falhas passa a logaritmo. Através do comportamento da curva de performance representadas nas Figuras 48 e 49, observa-se que a performance do sistema centralizado é mais sensível ao aumento da complexidade do que a performance do sistema distribuído.

No entanto, o sistema centralizado consegue uma performance melhor na rede de testes 2, a partir da complexidade 3, comparativamente à sua performance na rede de testes 1. A razão para tal deve-se ao elevado número de componentes que entram em falha na rede testes 2, a partir dos testes com os valores de complexidade anteriormente referidos. Nesta situação, a aprendizagem efectuada resulta na informação em como todos os componentes propagam falhas para todos os componentes, o que leva a uma maior performance comparativamente com a rede de testes 1.

De salientar, que o sistema distribuído mesmo em cenários caóticos consegue uma performance acima dos 60%, como é o caso da rede de testes 2 com uma complexidade 9, onde quase todos os componentes da rede, devido à elevada conectividade, entram em falha devido á propagação.

6. CONCLUSÕES E TRABALHO FUTURO

Face á evolução dos sistemas de produção, em grande parte devido ao surgimento de novas tecnologias e ao aumento dos requisitos de produção, a vertente de diagnóstico tem ganho cada vez mais importância. Um sistema de diagnóstico moderno e eficiente não é apenas visto como um método para identificar e analisar falhas, mas é também visto como um pilar fundamental de novos sistemas de produção como os EPS. Apresentando capacidades de evolução e adaptação, os sistemas de diagnóstico podem desempenhar um papel fundamental em acções de prevenção, manutenção, prognóstico de falhas, etc.

No entanto, o desenvolvimento de sistemas de diagnóstico, que apresentem as características referidas anteriormente, e sejam aplicáveis aos EPS torna-se difícil. As características inerentes a estes sistemas, como a quantidade e a natureza do tipo de informação trocada através das múltiplas interacções entre os componentes do sistema, tornam mais difícil a identificação de falhas e as suas propagações.

Neste sentido, torna-se interessante abordar a problemática de diagnóstico tendo como foco as interacções entre os vários componentes, pelas quais se podem propagar falhas. Nesta tese foi feita uma comparação entre dois sistemas de diagnóstico que modelam a informação do sistema numa perspectiva de rede de interacções e realizam o diagnóstico tendo essas interacções como base. Através desta metodologia, a especificidade dos componentes do sistema passa a ser quase abstracta para o diagnóstico, já que este baseia-se nas interacções que os componentes estabelecem entre si e não nos detalhes dos próprios componentes.

Através da performance obtida pelos sistemas de diagnóstico pode-se concluir que num sistema mais estruturado e mais clássico como é o caso da NOVAFLEX, onde a disposição dos componentes está de certo modo organizada, ambos os sistemas de diagnóstico obtém resultados muito satisfatórios.

No entanto quando os sistemas de diagnóstico são aplicados em redes de componentes geradas pela plataforma de testes, os resultados obtidos são esclarecedores em relação ao potencial de cada um dos diagnósticos. Face a estes

resultados, pode-se concluir que o sistema distribuído, ao contrário do centralizado e perante cenários de falhas, consegue resultados muito promissores.

No futuro, e face aos resultados obtidos, torna-se legítimo afirmar que as tomadas de decisões possam ser feitas localmente, ou seja, que o processo de diagnóstico seja implementado de uma forma distribuída e embutida nos próprios componentes.

No entanto, a abordagem que ambos os sistemas de diagnóstico fazem, do ponto de vista de uma rede constituída por componentes e suas interacções entre componentes, é inovadora e representa um passo importante na modelação e abstracção dos respectivos componentes que é necessário para efectuar conseguir realizar um processo de diagnóstico.

Durante a realização deste trabalho resultaram três artigos científicos, que validam o estudo efectuado e os resultados obtidos, sendo submetidos e aceites nas seguintes conferências internacionais:

- "Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems." [73]
- "15th IEEE International Conference on Emerging Technologies and Factory Automation." [74]
- "Second edition of the Doctoral Conference on Computing, Electrical and Industrial Systems, DoCEIS'11." (submetido)

Um dos desafios, no seguimento do estudo entre dois diagnósticos que seguem uma abordagem centralizada e uma distribuída, é até que ponto o nível de abstracção da informação, utilizada para modelar o problema numa perspectiva de rede, pode ter um impacto nos resultados alcançados. Ou seja, que níveis de granularidade podem ser utilizados para abordar o problema e que efeitos terá isso do ponto de vista da performance dos sistemas de diagnóstico.

7. REFERÊNCIAS

[1] P. Érdi, “Complexity explained.”, Berlin: Springer Verlag, 2008.

[2] K. Ueda, "A concept for bionic manufacturing systems based on DNA-type information", in *PROLAMAT* Tokyo: IFIP, 1992.

[3] L. Gou, P. B. Luh, and Y. Kyoka, "Holonic Manufacturing Scheduling Architecture, Cooperation Mechanism and Implementation." *Computers in Industry,* vol. 37, pp. 213-231, 1998.

[4] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts, and P. Peeters, "Reference architecture for holonic manufacturing systems: PROSA." *Computers in Industry,* vol. 37, pp. 255-274, 1998.

[5] S. Bussmann and D. C. Mcfarlane, "Rationales for Holonic Manufacturing." in *Second International Workshop on Intelligent Manufacturing Systems*, Leuven, Belgium, pp. 177 – 184, 1999.

[6] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritchow, A. G. Ulsoy, and H. Van Brussel, "Reconfigurable Manufacturing Systems." *CIRP Annals - Manufacturing Technology,* vol. 48, pp. 527-540, 1999.

[7] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing Systems and their Enabling Technologies." *International Journal Manufacturing Technology and Management,* vol. 1, pp. 113-130, 2000.

[8] M. Onori, "Evolvable Assembly Systems - A New Paradigm?" in *33rd International Symposium on Robotics* Stockholm, 2002.

[9] Barata J., Onori M., Frei R., Leitão P.; “Evolvable Production Systems: Enabling Research Domains.”, 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production. CARV, Toronto Canada, 2007

[10] L. Ribeiro, J. Barata, G. Cândido, and M. Onori, "Evolvable Production Systems: An Integrated View on Recent Developments." in *6th International Conference on Digital Enterprise Technology* Hong Kong: IEEE, 2009.

[11] O. Holland and C. Melhuish, "Stigmergy, self-organization, and sorting in collective robotics," *Artificial Life,* vol. 5, pp. 173-202, 1999.

[12] Maffei A., Onori M., “A Preliminary Study of Business Model for Evolvable Production Systems.” in IEEE/ISAM conference Seoul Korea. 2009.

[13] R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: a survey." *Journal of Intelligent Manufacturing,,* vol. 17, pp. 111–131, 2006.

[14] Gruver, W. A., D. B. Kotak, “Holonic Manufacturing Systems: Phase II.” HoloMAS**:** pp. 1-14, 2003.

[15] Bowers, C.P.; “Formation of Modules in a Computational Model of Embryogeny.”, Proceedings of the 2005 Congress on Evolutionary Computation, Vol. 1, 2-5, pp.537-542, September 2005.

[16] E. Hoda, "Flexible and reconfigurable manufacturing systems paradigms." *International Journal of Flexible Manufacturing Systems (Special Issue: Reconfigurable Manufacturing Systems),* vol. 17, pp. 261-276, October, 2006.

[17] Onori M., Alsterman H., “An architecture development approach for evolvable assembly systems.”, 2005.

[18] Barata, J., P. F. Santana, “Evolvable Assembly Systems: A Development Roadmap.”, IFAC, 2006.

[19] Onori, M., J. Barata, “Evolvable Assembly Systems Basic Principles.”, BASYS, 2006

[20] Regina Frei, Luis Ribeiro, José Barata, Daniel Semere, “Evolvable Assembly Systems: Towards User Friendly Manufacturing.”, 2007.

[21] Barata, J., R. Frei, “Evolvable Production Systems Context and Implications.”, 2007.

[22] Frei, R., Barata, J., and Di Marzo Serugendo, G.,"A complexity theory approach to evolvable production systems.", Internacional Conferecene in informatics and control, automation and robotics, Angers, France, 2007.

[23] Bengt Lindberg, Mauro Onori, Daniel T. Semere, “Evolvable Production Systems - A Position Paper”, 2007.

[24] Barata, J. and M. Onori, “Evolvable Assembly and Exploiting Emergent Behaviour.”, ISIE, 2006.

[25] Wooldridge, M., and Jennings, N. R., "Intelligent Agents - Theory and Practice.", *Knowledge Engineering Review*, 10(2), 115-152, 2005.

[26] Wooldridge, M. J., and Jennings, N. R., "Agent Theories, Architectures, and languages: A Survey." *ECAI-Workshop on Agent Theories, Architectures and Languages*, Amsterdam, 1-32, 2005.

[27] J. Barata, L. Ribeiro, “Diagnosis on Evolvable Production Systems.”, International Symposium on Industrial Electronics. Vigo, IEEE, 2007.

[28] P. Leitao and F. Restivo, "Agent-based Holonic Production Control." presented at 13th International Workshop on Database and Expert Systems Applications, 2002.

[29] J. Lastra, "Reference Mechatronic Architecture for Actor-Based Assembly Systems, PhD thesis.", Tampere, 2004.

[30] J. Barata, "Coalition Based Approach for Shop Floor Agility, PhD thesis.", Monte da Caparica, 2003.

[31] EUPASS, "Evolvable Ultraprecision Assembly Systems." http://www.eupass.org/, 2006.

[32] SODA, "Service Oriented Device and Delivery Architecture.", http://www.soda-itea.org/Home/default.html, 2006.

[33] SIRENA, "Service Infrastructure for Real-time Embedded Network Applications.", http://www.sirena-itea.org/Sirena/Home.htm, 2006.

[34] SOCRADES, "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices." http://www.socrades.eu/Documents/AllDocuments/default.html, 2006.

[35] Luis Ribeiro, Jose Barata, Armando Colombo, (2007), “MAS and SOA: A Case Study Exploring Principles and Technologies to Support Self-Properties in Assembly Systems.”, Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, 2008.

[36] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1 W3C Note 15 March 2001.", 2001.

[37] V. Venkatasubramanian*, et al.*, "A review of process fault detection and diagnosis: Part I: Quantitative model-based methods.", *Computers & Chemical Engineering,* vol. 27, pp. 293-311, 2003.

[38] V. Venkatasubramanian*, et al.*, "A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies.", *Computers & Chemical Engineering,* vol. 27, pp. 313-326, 2003.

[39] V. Venkatasubramanian*, et al.*, "A review of process fault detection and diagnosis: Part III: Process history based methods.", *Computers & Chemical Engineering,* vol. 27, pp. 327-346, 2003.

[40] J. Gertler, "Fault detection and isolation using parity relations.", *Control Engineering Practice,* vol. 5, pp. 653-661, 1997.

[41] G. Welch and G. Bishop, "An introduction to the Kalman filter.", *University of North Carolina at Chapel Hill, Chapel Hill, NC,* 1995.

[42] S. Lapp, "Computer-aided synthesis of fault-trees." *IEEE Transactions on Reliability,* 1977.

[43] Mobley, R. K., *“An Introduction to Predictive Maintenance.”* Butterworth-Heinemann, 2002.

[44] A. K. S. Jardine, D. Lin, and D. Banjevic, "A review on machinery diagnostics and prognostics implementing condition-based maintenance." MECHANICAL SYSTEMS AND SIGNAL PROCESSING, vol. 20, pp. 1483-1510, 2006.

[45] A. H. C. Tsang, "Condition based maintenance: tools and decision making.", Journal of Quality in Maintenance Engineering, vol. 1,pp. 3-17, 1995.

[46] Lopes, L. F. d. S., "Robot Learning at Task Level," New University of Lisbon, Lisbon, 1997.

[47] Olsson, E., Funk, P., and Xiong, N., "Fault Diagnosis in Industry Using Sensor Readings and Case-Based Reasoning.", *Journal of Intelligent & Fuzzy Systems*, 15, 2004.

[48] McIntyre, M. L., Dixon, W. E., and Walker, I. D., "Fault Identification for Robot Manipulators.", *IEEE TRANSACTIONS ON ROBOTICS*, 21(5), 2005.

[49] Tinós, R., and Terra, M. H., "Fault detection and isolation for multiple manipulators.", 15th IFAC World Congress, Barcelona, Spain, 2002.

[50] Drif, M. h., and Cardoso, A. M., "Rotor Cage Fault Diagnostics in Three-Phase Induction Motors, by the Instantaneous Non-Active Power Signature Analysis.", International Symposium on Industrial Electronics, IEEE, Vigo, 2007.

[51] Zhang, S., Asakura, T., Xu, X., and Xu, B. "Fault Diagnosis System for Rotary Machines Based on Fuzzy Neural Networks.", *International Conference on Advanced Intelligent Mechatronics*, 199- 204, 2003.

[52] Barigozzi, A., Magni, L., and Scatollini, R., "A Probabilistic Approach to Fault Diagnosis in Industrial Systems.", *IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY*, 12(6), 2004.

[53] Zhou, S., Zhang, J., and Wang, S., "Fault diagnosis in industrial processes using principal component analysis and hidden markov models.", American Control Conference, Boston, USA, 2004.

[54] Rodrigues, M. A., Liu, Y., Bottaci, L., and Rigas, D. I., "Learning and diagnosis in manufacturing processes through an executable Bayesian network.", *13th international conference on Industrial and engineering applications of artificial intelligence and expert systems: Intelligent problem solving: methodologies and approaches*, New Orleans, Louisiana, United States, 390 – 395, 2003.

[55] Balduzzi, F., and Febbraro, A. D., "Combining fault detection and process optimization in manufacturing systems using first order hybrid petri nets.", International Conference on Robotics & Automation, IEEE, Seoul, 40-45, 2001.

[56] Cárdenas, C., Olmos, J., García, D., and Baeyens, E., "Modelling, supervision and diagnosis of a manufacturing cell.", Emerging Technologies and Factory Automation, IEEE, Lisbon, Portugal, 2003.

[57] L.P. Khooa, C.L. Angb, J. Zhangc, "A fuzzy-based genetic approach to the diagnosis of manufacturing systems.", Engineering Applications of Artificial Intelligence 13, 303-310, 2000.

[58] W. Hu, A.G. Starr, A.Y.T Leung, "Operational fault diagnosis of manufacturing systems.", Journal of Material Processing Technology 133 108-117, 2003.

[59] Fries, T. P., and Graham, J. H., "Fusion of soft and hard computing for fault diagnosis in manufacturing systems.", *International Conference on Systems, Man and Cybernetics*, Washington D. C., USA, 114 – 119, 2003.

[60] Luo, J., Pattipati, K. R., Qiao, L., and Chigusa, S., "Agent based real time fault diagnosis.", *Aerospace Conference*, 3632- 3640, 2005.

[61] Klein, F., and Tichy, M., "Building reliable systems based on self-organizing multiagent systems.", *International workshop on Software engineering for large-scale multi-agent systems* Shanghai, China, 51 – 58, 2006.

[62] M. T. Long, R. R. Murphy, and L. E. Parker., "Distributed multi-agent diagnosis and recovery from sensor failures.", In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003.

[63] Nico Roos, Annette ten Teije, André Bos and Cees Witteveen., "An analysis of MultiAgent Diagnosis.", AAMAS'02, 15-19, July, Bologna Italy, 2002.

[64] Nico Roos, Annette ten Teije, and Cees Witteveen., "A Protocol for Multi-Agent Diagnosis with Spatially Distributed Knowledge.", AAMAS'03, July 14–18, Melbourne Australia, 2003.

[65] L. Ribeiro, J. Barata, and J. Ferreira, "The Meaningfulness of Consensus and Context in Diagnosing Evolvable Production Systems," in *Emerging Trends in Technological Innovation*. vol. 1, L. M. Camarinha-Matos, P. Pereira, and L. Ribeiro, Eds. Berlin: Springer, 2010.

[66] JADE, (2007), "Java Agent Development Framework.", http://www.jade.tilab.com.

[67] JUNG, (2003), "Java Universal Network/Graph Framework", http://jung.sourceforge.net.

[68] Drools, Version 5.0, (2009), http://jboss.org/drools.

[69] http://downloads.jboss.com/drools/docs/5.1.1.34858.FINAL/drools-expert/html_single/index.html

[70] http://www.fipa.org/specs/fipa00028/SC00028H_files/image001.gif

[71] L. Ribeiro, J. Barata, and J. Ferreira, “An Agent-Based Interaction-Oriented Shop Floor to Support Emergent Diagnosis.”, IEEE International Conference on Industrial Informatics, Osaka, Japan, 2010.

[72] S.N. Dorogovtsev and J.F.F. Mendes, “Evolution of Networks: From Biological Nets to the Internet and WWW”, 2003.

[73] Luís Ribeiro, José Barata, Bruno Alves and João Ferreira, “Global vs Local: A Comparison of two Approaches to Perform Diagnosis in Networks of Mechatronic Agents.”, “Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems”, Budapeste, Hungria, 2010.

[74] Luís Ribeiro, José Barata and Bruno Alves, “Exploring the Network Dimension of Diagnosis in Evolvable Production Systems.”, “15th IEEE International Conference on Emerging Technologies and Factory Automation”, Bilbao, Espanha, 2010.

[75] Danail Bonchev and Gregory A. Buck, “Quantitative Measures of Network Complexity”, chapter 5 in “Complexity in Chemistry, Biology, and Ecology”, 2005.

[76] Watts D.J., “A simple model of global cascades on random networks”, Proceedings of the National Academy of Sciences, Volume 99, p.5766-5771, 2002.

[77] Kai F. Goebel, Paul K. Wright, “Monitoring and Diagnosing Manufacturing Processes Using a Hybrid Architecture with Neural Networks and Fuzzy Logic.”, EUFIT '93 - First European Congress on Fuzzy and Intelligent Technologies, Aachen, 1993.

[78] James Kurien, Xenofon Koutsoukos, Feng Zhao, “Distributed Diagnosis of Networked, Embedded Systems.”, In Proceedings of the 13th International Workshop on Principles of Diagnosis, 2002.

[79] Barata J., Ribeiro L., Colombo A., “Diagnosis using Service Oriented Architectures (SOA).”, Industrial Informatics, 5th IEEE International Conference, 2007.

[80] Luis Ribeiro, José Barata, Paulo Leitão, Neslon Silvério, “Maintenance Management and Operational Support as Services in Reconfigurable Manufacturing Systems”, IFAC Symposium on Information Control Problems in Manufacturing. 13th. Moscow, 2009.

[81] Pedro Neves, Mauro Onori, “Evolvable Production Systems: Approach towards Modern Production Systems.“, Proceedings of DET2009 6th International Conference on Digital Enterprise Technology Hong Kong, 2009.

[82] J. Barata, M. Onori, R. Frei, P. Leitão, “Evolvable Production Systems in a RMS Context: Enabling Concepts and Technologies.”, Proceedings of the 2nd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV’07), 2007.

[83] Onori, M., J. Barata, “Evolvable Assembly Systems Basic Principles.”, Conference on Information Technology for BALANCED AUTOMATION SYSTEMS in Manufacturing and Services. Ontario, Canada, Springer, 2006.

[84] Hu W., Starr A.G., Zhou Z., Leung A.Y.T., “A systematic approach to integrated fault diagnosis of flexible manufacturing systems.”, International Journal of Machine Tools and Manufacture, Volume 40, Number 11, September 2000 , pp. 1587-1602(16).

[85] Dash S., Venkatasubramanian V., “Challenges in the industrial applications of fault diagnostic systems.”, Computers and Chemical Engineering, Volume 24, Number 2, 15 July 2000 , pp. 785-791(7).

[86] Paulo Leitão, “Agent-based distributed manufacturing control: A state-of-the-art survey.”, Engineering Applications of Artificial Intelligence, Volume 22, pp. 979-991, 2009.

[87] M. G. Mehrabi, A. G. Ulsoy, Y. Koren, P. Heytler, “Trends and perspectives in flexible and reconfigurable manufacturing systems”, Journal of Intelligent Manufacturing, Vol. 13, No. 2. (1 April 2002), pp. 135-146.

[88] Weiming Shen, Qi Hao, Hyun Joong Yoon, Douglas H. Norrie, “Applications of agent-based systems in intelligent manufacturing: An updated review.”, Advanced Engineering Informatics 20 (2006) pp. 415-431.

[89] João Ferreira, “Diagnosis of an EPS module.”, Master degree Thesis in Electrical and Computer Engineering, 2010.